# New Foundations for Efficient Authentication, Commutative Cryptography, and Private Disjointness Testing

by

Stephen A. Weis

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2006



Author .....
Department of Electrical Engineering and Computer Science
May 25, 2006

Certified by .....
Ronald L. Rivest
Viterbi Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

# New Foundations for Efficient Authentication, Commutative Cryptography, and Private Disjointness Testing

by

Stephen A. Weis

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

## Abstract

This dissertation presents new constructions and security definitions related to three areas: authentication, cascadable and commutative crytpography, and private set operations. Existing works relevant to each of these areas fall into one of two categories: efficient solutions lacking formal proofs of security or provably-secure, but highly inefficient solutions. This work will bridge this gap by presenting new constructions and definitions that are both practical and provably-secure.

The first contribution in the area of efficient authentication is a provably-secure authentication protocol named HB$^+$. The HB$^+$ protocol is efficient enough to be implemented on extremely low-cost devices, or even by a patient human with a coin to flip. The security of HB$^+$ is based on the hardness of a long-standing learning problem that is closely related to coding theory. HB$^+$ is the first authentication protocol that is both practical for low-cost devices, like radio frequency identification (RFID) tags, and provably secure against active adversaries.

The second contribution of this work is a new framework for defining and proving the security of cascadable cryptosystems, specifically commutative cryptosystems. This new framework addresses a gap in existing security definitions that fail to handle cryptosystems where ciphertexts produced by cascadable encryption and decryption operations may contain some message-independent *history*. Several cryptosystems, including a new, practical commutative cryptosystem, are proven secure under this new framework.

Finally, a new and efficient private disjointness testing construction named HW is offered. Unlike previous constructions, HW is secure in the face of malicious parties, but without the need for random oracles or expensive zero-knowledge protocols. HW is as efficient as previous constructions and may be implemented using standard software libraries. The security of HW is based on a novel use of *subgroup assumptions*. These assumptions may prove useful in solving many other private set operation problems.

Thesis Supervisor: Ronald L. Rivest
Title: Viterbi Professor of Electrical Engineering and Computer Science

# Acknowledgments

Thanks to...

...my family Joseph, Elaine, Erica, and Sophie Weis for their love and support.

...my advisor Ronald Rivest for his help and guidance.

...my thesis committee members Srini Devadas and Silvio Micali.

...my co-authors Ari Juels and Susan Hohenberger.

...Shafi Goldwasser and Sanjay Sarma for their input.

...Ben Adida and Rafeal Pass for their help.

...Burt Kaliski and RSA Security for providing research opportunities.

...my friends Michael Axlegaard, Mathias and Lorelei Craig, Belle Darsie, Lawrence Gold, Chris Gorog, Pawel Krewin, Mycah Mattox, Claire Monteleoni, Chudi Ndubaku, Abhi Shelat, and anyone else I forgot to mention.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The pervasive deployment of low-cost devices and wireless networks is making many new and exciting applications viable. Data once stored in centralized databases and transmitted over fixed networks is becoming available anywhere at any time. Low-cost devices are increasingly embedded in physical objects as identifiers, creating a bridge between the digital and physical worlds.

Yet, despite creating new opportunities, these burgeoning technologies may also create new risks to privacy and security. Data once controlled by single entities or stored in closed systems are increasingly distributed among different parties, platforms, and network mediums. These varied systems must contend with noisy, unreliable environments as well as malicious adversaries and untrusted channels.

While existing security and cryptographic techniques may address many security and privacy issues in traditional settings, they often make assumptions that do not hold in a highly-pervasive or *ad hoc* settings. For instance, parties are often assumed to have ample computational power, which is not the case in many low-cost systems.

A compelling example of such a technology are radio frequency identification (RFID) systems. These systems consist of simple, low-cost *tags* that are attached to physical objects, and more powerful *readers* that wirelessly access data stored on tags. By binding digital data to physical objects, RFID tags enable extremely useful automatic identification systems.

Billions of RFID tags are already deployed. Tens of billions may be deployed in the near future, making RFID tags the most pervasive microchip in history. Other pervasive systems like sensor networks may likewise experience rapid growth.

These types of pervasive devices will be used in applications as far-ranging as supply-chain management, physical access control, payment systems, environmental sensors, and anti-counterfeiting. Consumers, retailers, transporters, manufacturers, and government agencies will all be affected by the widespread adoption of these devices.

All manner of sensitive data may be stored, collected, and inferred from these types of pervasive systems. For instance, passports, currency, prescription drugs, clothing, or military matériel may all contain pervasive devices storing sensitive data. Without proper security controls, this data might be abused and threaten everything from personal privacy to national security.

Often, pervasive devices will have only hundreds of gates available for security features. These weak devices will often have no battery, no clock, and no way to communicate except through other possibly untrusted devices. This begs the question: Can such feeble devices offer any notion of security?

One might argue that advancements in fabrication and manufacturing will render this question moot in a matter of years. According to Moore's law, won't these devices be twice as powerful in a couple years? For widely pervasive, low-cost devices, the answer is no.

Widespread pervasive systems will be under extreme economic pressures. Unlike personal computers, devices like RFID will be produced in huge quantities and will have extremely low profit margins. Any additional circuitry must have immediate economic justification. Unless there is a compelling reason, the vast majority of purchasers will opt for the cheapest alternative.

Another issue is that future pervasive technologies may be built from printed organic components [110, 111]. Early organic components have a much lower gate density than traditional silicon components, and will thus require a much larger surface area. Physical constraints in many pervasive applications will translate into a tight limit on the gates available in an organic circuit. Since organic circuit technology is so young, printed circuits will have low gate densities for years to come.

There are several interesting security issues in pervasive and wireless systems. One issue is that environments may be populated with devices belonging independent systems, as well as malicious adversaries. In many situations, independent, but mutually suspicious, parties may have a strong interest to correlate some data without sharing all data. For instance, independent or competing environmental sensor networks may each benefit by sharing some of their data.

This type of problem falls under the general category of private set operations or privacy-preserving data mining. Private set operation problems arise in both traditional settings as well as pervasive systems. For instance, suppose a reader detects some unknown tag and wishes to establish whether it "owns" that particular tag. Neither party can broadcast identifiers in the clear, otherwise an eavesdropper could collect and correlate data about that system.

This problem may be viewed as a private intersection, private intersection cardinality, or private disjointness test. That is, given a reader's database $R$ and a tag's set of identities $T$, the reader wishes to respectively learn $R \cap T$, $|R \cap T|$, or $|R \cap T| > 0$ without leaking any information to eavesdroppers.

As another example of this type of private set operation problem in a more general computing setting, imagine a law-enforcement agency has a list of suspects $S$ and wishes to determine whether any member of $S$ is among a list of passengers $P$ on a flight. The law-enforcement agency cannot reveal $S$ without compromising its investigation, while the airline cannot reveal $P$ (without subpoena) due to privacy regulations. However, both parties have a strong interest in notifying law-enforcement whether $|P \cap S| > 0$.

Again, there are many existing solutions to these types of problems, since they can be viewed as general secure multi-party computation problems. Yet, traditional solutions are often too inefficient to be used in practice, even on general purpose

personal computers.

Several solutions that might be practical make use of commutative cryptosystems. These cryptosystems allow several parties to encrypt messages under different keys in a cascade of operations, then decrypt the ciphertext under the same set of keys in an arbitrary order. Several "folklore" commutative cryptosystems have existed for over 20 years, and form the basis of many applications and protocols.

Surprisingly, very little has been said about the security of either these specific constructions, or of commutative cryptosystems in general. For instance, even basic notions of semantic security may fail to hold if a ciphertext reveals some message-independent *history* about the sequence of operations that produce it. Despite being a useful and intuitive tool, commutative cryptosystems, and in fact, cascadable cryptosystems in general, has been overlooked by traditional security definitions.

**Organization:** This work will address three of the aforementioned security issues and present new contributions in each area. Chapter 2 develops a new, efficient, provably-secure authentication protocol named $\text{HB}^+$ that is appropriate for extremely low-cost devices. Chapter 3 presents a new definitional framework for proving the security of cascadable and commutative cryptosystems, as well as a new, efficient commutative cryptosystem construction. Finally, Chapter 4 presents a private disjointness testing construction that is secure against a stronger class of adversaries than prior work for equivalent computational costs, and without the use of random oracles, bilinear maps, or expensive zero-knowledge proofs.

## 1.1 Efficient Authentication

Chapter 2 affirmatively answers the question "*Can feeble devices authenticate themselves?*" by presenting a new, efficient authentication protocol dubbed $\text{HB}^+$. The $\text{HB}^+$ protocol is the first authentication protocol secure against active adversaries that is efficient enough to be implemented in pervasive devices like RFID tags. Work on $\text{HB}^+$ was conducted with Ari Juels of RSA Security and originally appeared in [67].

$\text{HB}^+$efficiently addresses a security vulnerability in a protocol due to Hopper and Blum (HB) [61, 62]. Although secure against passive eavesdroppers, HB critically fails in the presence of active adversaries able to initiate their own protocols. The HB protocol was originally intended for human-to-computer authentication, similar to earlier protocols due to Matsumoto and Imai [79, 80, 118]. This author originally observed the similarities between the human-computer and to the pervasive computing setting in [121].

Chapter 2 will improve the concrete security bounds of the original HB work [62] and prove $\text{HB}^+$ secure under the same assumptions. It will also consider variants of the $\text{HB}^+$ protocol, such as a parallel version proven secure by Katz and Shin [69], and a two-round version, whose security is an open question.

The HB protocol's underlying hardness is based on the "Learning Parity with Noise" (LPN) problem. This problem is closely related to the problem of decoding

random linear codes [76, 50] and has been the basis of several cryptosystems over the years [8, 91, 108, 24, 109, 28].

The LPN problem is the subject of both ongoing complexity research [10, 70, 116, 11, 100] and practical algorithmic design research [119, 63, 54]. The LPN problem is a viable alternative hardness assumption to factoring or finding discrete logarithms. Practical implementation costs and key lengths will be analyzed in Chapter 2. Developing practical attacks against $HB^+$ may ultimately improveme the best known algorithm for solving the LPN problem [11]. In fact, subsequent works motivated by $HB^+$ have already improved the constant factors of the best known asymptotic algorithm.

## 1.2 Commutative Cryptography

Chapter 3 answers the question "*How can you prove the security of a commutative cryptosystem?*" by defining a new security framework that is compatible with cascadable and commutative cryptosystems. This chapter incorporates a flexible string rewrite system model of cascaded cryptographic operations into a new notion of *cascadable semantic security*. A new notion of *historical security* is also introduced. Several cryptosystems are proven secure under the new model, including a provably-secure and efficient commutative cryptosystem. The work in this chapter was conducted with Ronald Rivest and appears in [102].

Several private set operation schemes [27, 1] make use of cascaded or commutative cryptographic operations. The classic Shamir Three-Pass protocol also relies on commutativity. These applications typically apply on the well-known commutative Pohlig-Helman [99] and Massey-Omura [78] encryption schemes. These cryptosystems have the property that one may sequentially encrypt a message under several different keys, then decrypt in an arbitrary order.

Both Pohlig-Helman and Massey-Omura lack formal proofs of security. In fact, in working to define a secure commutative encryption scheme that might be used for set intersection or data mining applications, one finds that standard security definitions are fundamentally incompatible with cascadable and commutative cryptosystems.

For instance, a standard indistinguishability under chosen plaintext attack (IND-CPA) experiment does not accommodate commutative cryptosystems. This is because ciphertexts may have some message-independent *history* about the cascaded sequence of operations that produced them. Ciphertexts may be trivially distinguished by their history, although nothing about the underlying message is revealed. Thus, an otherwise valid cascadable cryptosystems would not be semantically secure in the traditional model.

The same applies to indistinguishability under adaptive chosen-ciphertext attacks (IND-CCA). In that case, an adversary with access to a decryption oracle may trivially distinguish ciphertexts in commutative cryptosystems, without learning any information about underlying messages.

Chapter 3 addresses these definitional deficiencies by proposing new, generalized security definitions that accommodate commutative properties. To do so, it will first

model cryptographic operations with a string rewrite system [34, 32]. Strings of symbols will represent a cascaded sequence of operations. Rewrite rules will model the effects of cryptographic operations. Using this type of string rewrite system has been used in analysis of cryptographic protocols by treating messages as terms in the rewrite system [48, 86, 82, 83].

Chapter 3 defines a more general notion of semantic security based on the underlying string rewrite system model. Basically, one will be able to "plug in" a string rewrite system into the security definition that meets certain properties. A result is that these new definitions will be both "backwards compatible" with standard security definitions, and may be compatible with more complex string or term rewrite systems modeling re-randomization or homomorphic operations.

Finally, Chapter 3 will present a new commutative encryption scheme based on Golle et al.'s universally re-encryptable cryptosystem [53]. This commutative encryption scheme allows multiple parties to sequentially encrypt a ciphertext, then decrypt in an arbitrary order. This scheme is efficient and requires only standard modular exponentiation operations. It will be proved secure using the new string rewrite-based security definitions presented in this section.

## 1.3 Private Set Operations

Chapter 4 answers the question "*Can two parties efficiently and privately determine whether they share any values?*" by presenting a new, efficient private disjointness testing construction. This construction is secure against a stronger class of adversaries than previous constructions, yet requires equivalent computation. Furthermore, this construction will require no random oracles, bilinear maps, nor any expensive zero-knowledge protocols. Work in Chapter 4 was conducted with Susan Hohenberger (thus will be referred to as "HW") and appears in [59].

Besides general secure multi-party computation techniques, there are many existing private set operations protocols, for instance works by Pinkas, Naor, and Lindell, Freedman, and Nissim [90, 75, 98, 46]. A particular application of private set operations is in privacy-preserving data mining [75, 27, 1].

The Freedman, Nissim, Pinkas (FNP) scheme [46] offers a very useful design paradigm for private set operations. The FNP format is the basis of both the Kiayias and Mitrofanova (KM) [71] protocol and HW. However, both FNP and KM suffer a fundamental security flaw: it is trivial for one malicious party to convince the other that an intersection exists.

Both FNP and KM address this problem, although they must use random oracles, universally-composable commitments, repeated invocations, or expensive zero-knowledge proofs to secure their constructions against malicious adversaries. The HW construction is secure against a single malicious party without any additional computation.

FNP uses Paillier's homomorphic encryption scheme [94, 95, 23] as an underlying operation. Kiayias and Mitrofanova rely on a homomorphic ElGamal variant first used in voting schemes by Cramer, Gennaro, and Schoenmakers [29]. In contrast, the HW

construction will make use of a new "testable and homomorphic commitment" (THC) primitive that is related to both Pedersen commitments [96] and Boneh, Goh, Nissim's (BGN) small-message encryption [13]. We offer an efficient THC construction based on subgroup assumptions, but does not make use of bilinear maps.

By simply replacing Paillier encryption with THCs, a FNP-style protocol will be naturally secure against one malicious party with no additional security assumptions. The THC construction is simple to implement using standard software libraries and offers equivalent performance to existing private disjointness testing protocols. Testable and homomorphic commitments could have useful applications in other settings as well.

# Chapter 2

# Efficient Authentication

Forgery and counterfeiting are emerging as serious security risks in low-cost pervasive computing devices. Authentication could address many of these risks, yet traditional techniques are often much too costly for low-cost devices. These devices lack the computational, storage, power, and communication resources necessary for most cryptographic authentication schemes.

Low-cost Radio Frequency Identification (RFID) tags are examples of a pervasive, yet resource-constrained device. Since low-cost devices like RFID are major beneficiaries of this work, this chapter will use RFID tags as a motivating example for discussion of issues surrounding low-cost authentication. However, none of the results presented in this work are RFID-specific; the protocols presented in this chapter could be implemented in general computing settings – or even by a human with a coin to flip and time to spare.

Low-cost RFID tags in the form of Electronic Product Codes (EPC) are poised to become the most pervasive device in history [40]. Already, there are billions of RFID tags on the market, used for applications like supply-chain management, inventory monitoring, access control, and payment systems [101, 123]. Proposed as a replacement for the Universal Product Code (UPC) (the barcode found on most consumer items), EPC tags are likely one day to be affixed to everyday consumer products.

Today's generation of basic EPC tags lack the computational resources for strong cryptographic authentication. These tags may only devote hundreds of gates to security operations. EPC tags often passively harvest power from radio signals emitted by tag readers. This means they have no internal clock, nor can perform any operations independent of a reader.

In principle, standard cryptographic algorithms – asymmetric or symmetric – can support authentication protocols. Implementing an asymmetric cryptosystem like RSA in EPC tags is entirely infeasible. RSA implementations require tens of thousands of gate equivalents. Even the storage for RSA keys would dwarf the memory available on most EPC tags.

Standard symmetric encryption algorithms, like DES or AES, are also too costly for EPC tags. While current EPC tags may have at most 2,000 gate equivalents available for security (and generally much less), common DES implementations require tens of thousands of gates. Although recent light-weight AES implementations

require approximately 5,000 gates [43], this is still too expensive for low-cost EPC tags likely to be deployed within in the next five to ten years.

It is easy to brush aside consideration of these resource constraints. One might assume that Moore's Law will eventually enable RFID tags and similar devices to implement standard cryptographic primitives like AES. But there is a countervailing force: Many in the RFID industry believe that pricing pressure and the spread of RFID tags into ever more cost-competitive domains will mean little effective change in tag resources for some time to come, and thus a pressing need for new lightweight primitives.

This is especially true in organic printed circuits for RFID [111, 110]. Early organic components are physically much large than their silicon counterparts. Although, organic circuits may eventually be much cheaper to produce than traditional silicon circuits, they will have a much lower gate density. Since physical products may impose a maximum physical size on tags, organic-based RFID may necessarily have extremely low gate counts for the foreseeable future.

Surprisingly, low-cost pervasive devices like Radio Frequency Identification (RFID) tags share similar capabilities with another weak computing device: people. These similarities motivate the adoption of techniques from human-computer security to the pervasive computing setting. This chapter analyzes a particular human-to-computer authentication protocol designed by Hopper and Blum (HB), and shows it to be efficient for low-cost pervasive devices. This chapter offers an improved, concrete proof of security for the HB protocol against passive adversaries.

The main contribution of this chapter is an augmented version of the HB protocol, named HB$^+$, that is secure against active adversaries. The HB$^+$ protocol is a novel, symmetric authentication protocol with an efficient, low-cost implementation. This chapter proves the security of the HB$^+$ protocol against active adversaries based on the hardness of the Learning Parity with Noise (LPN) problem. Lack of security against active adversaries is a crucial flaw of HB that is efficiently addressed by this work. The HB$^+$ protocol presented in this chapter was originally published with Ari Juels of RSA Security in [67].

### 2.0.1 The Problem of Authentication

How does a computer or reader verify that the device it is communicating with is authentic? In the context of this chapter, "authentic" will mean that a device will share some secret with a computer, i.e. this work deals only with the symmetric key setting. A computer or reader will "own" a device if it knows the secret key stored on that device. This differs from the public-key setting, where devices might be authenticated using only public data.

It seems inevitable that many applications will come to rely on basic RFID tags or other low-cost devices as authenticators. For example, the United States Food and Drug Administration (FDA) proposed attaching RFID tags to prescription drug containers in an attempt to combat counterfeiting and theft [45]. These tags are supposed to serve two purposes. One is for inventory control and to detect thefts in the supply chain.

The second purpose is to provide a pedigree that allows pharmacists to trace prescription drugs back to their origin. The reason this is so important is that there is a massive market for counterfeit drugs. For some drugs, such as anti-malarials, nearly half of the drugs consumed are counterfeit, causing untold death and suffering [45]. Combined with tamper-evident packaging, RFID pedigrees will help ensure that the drugs consumed by end-users are real. However, there is an implicit assumption that RFID pedigrees may be authenticated and are difficult to forge.

Other RFID early-adopters include public transit systems and casinos. Several cities around the world use RFID bus and subway fare cards, and casinos are beginning to deploy RFID-tagged gambling chips and integrated gaming tables. Some people have even had basic RFID tags with static identifiers implanted in their bodies as payment devices or medical-record locators [115]. Again, the assumption is that it is difficult to forge these devices.

One heavily-criticized RFID application is the integration of RFID tags carrying biometric data in United States passports [114]. Although the primary concern is over personal privacy, forged RFID tags could be a national security risk. This is especially the case if electronic passports are assumed to be significantly harder to forge than traditional passports.

Today, most RFID devices simply broadcast a static identifier with no explicit authentication procedure. This allows an attacker to surreptitiously scan data needed to produce clones in what is called a *skimming* attack. An archetypal skimming setting might be an RFID-based subway pass system. An adversary might interrogate a device carried by someone riding a subway without detection. Skimming is obviously simple if tags broadcast fixed identifier values. Some *ad hoc* approaches for simple challenge-response protocols, such as XORing a challenge value with a fixed identifier, would crumble in the face of an active attacker.

Skimming opens the door to several other attacks. For example, in a *swapping* attack, a thief skims valid RFID tags attached to products inside a sealed container. The thief then manufactures cloned tags, seals them inside a decoy container (containing, e.g., fraudulent pharmaceuticals), and swaps the decoy container with the original. Thanks to the ability to clone a tag and prepare the decoy in advance, the thief can execute the physical swap very quickly. In the past, corrupt officials have sought to rig elections by conducting this type of attack against sealed ballot boxes [107].

Clones also create denial-of-service issues. If multiple, valid-looking clones appear in a system like a casino, must they be honored as legitimate? Or must they all be rejected as frauds? Cloned tags could be intentionally designed to corrupt supply-chain databases or to interfere with retail shopping systems. Denial of service is an especially critical threat to RFID-based military logistics systems.

Researchers have recently remonstrated practical cloning attacks against real-world RFID devices. Mandel, Roach, and Winstein demonstrated how to read access control proximity card data from a range of several feet and produce low-cost clones [77], despite the fact that these particular proximity cards only had a legitimate read range of several inches.

A team of researchers from Johns Hopkins University and RSA Laboratories re-

cently presented attacks against a cryptographically-enabled RFID transponder used automobile immobilization systems and a payment system called ExxonMobil SpeedPass [14]. SpeedPass allows customers at ExxonMobil service stations to purchase gas or other goods.

The JHU/RSA team was able to extract secret keys and simulate value transponders through an active skimming attack. Their attack exploited a weak, proprietary encryption scheme implemented on the underlying Texas Instruments RFID device, highlighting the relevance and importance of tag authentication. Millions of SpeedPass or automobile immobilization systems could be vulnerable to this type of attack.

### 2.0.2 Previous RFID Security Work

As explained above, a major challenge in securing RFID tags or other low-cost pervasive devices are their limited resources and small physical form. Table 2.1 offers specifications that might be realistic for near-future EPC tags. Such limited power, storage, and circuitry, make it difficult to implement traditional authentication protocols. This problem has been the topic of a growing body of literature.

A number of proposals for authentication protocols in RFID tags rely on the use of symmetric-key primitives. These works often assume cryptographically enhanced RFID tag functionality in the future, and do not propose use of any particular primitive. Other authors have sought to enforce privacy or authentication in RFID systems while avoiding the need for implementing standard cryptographic primitives on tags as well.

Two papers by Sarma, Weis, Rivest and Engels discuss security and privacy tools for RFID tags [104, 122]. In addition to privacy issues, they explore both tag-to-reader and reader-to-tag authentication protocols that rely on hash functions. This author also introduces the idea of adapting human authentication protocols to low-cost pervasive systems and highlights the HB protocol in [121].

Juels proposes "minimalist" cryptographic primitives that involve only XOR-based padding for authentication and privacy [65]. These work only in a limited adversarial model, however, and require writing of tags (a less reliable operation than reading). Juels also proposes "yoking proofs" [64] that can attest that pairs of tags were simultaneously scanned at some time. Yoking proofs do not require symmetric-key primitives, though, they do function only as one-time operations. Juels and Pappu discuss authentication issues arising in RFID-tagged currency [66]. In that specific setting, Juels and Pappu propose the use of optical information, such as a bar code, to enhance data integrity.

Henrici and Müller [57], and Ohkobu, Suzuki, and Kinoshita [92] present hash-based RFID privacy enhancements. Floerkemeier and Lampe discuss methods of implementing access control policies in RFID devices [44]. Feldhofer, Dominikus, and Wolkerstorfer [43] propose a low-cost AES implementation, potentially useful for higher-cost RFID tags, but still out of reach for basic tags in the foreseeable future.

Molnar and Wagner have considered RFID security and privacy issues in the library setting, highlighting security issues arising in other consumer applications [88]. Their authentication protocols rely on symmetric-key primitives, namely pseudo-

| | |
|---|---|
| **Storage:** | 128-512 bits of read-only storage |
| **Memory:** | 32-128 bits of volatile read-write memory |
| **Gate Count:** | 1000-10000 gates |
| **Security Gate Count Budget:** | 200-2000 gates |
| **Computation Clock Frequency:** | 868-956 MHz (UHF) |
| **Scanning Range:** | 3 meters |
| **Performance:** | 100 read operations per second |
| **Clock Cycles per Read:** | 10,000 clock cycles |
| **Tag Power Source:** | Passively harvested RF signal |
| **Power Consumption:** | 10 micro-watts |
| **Features:** | Anti-Collision Protocol Support<br>Random Number Generator |

Table 2.1: Example specification for a low-cost RFID tag

random functions. Notably, though, they address the important question of how to distribute tag-specific secrets to other principals in an RFID infrastructure. This is an oft-overlooked keystone of any authentication protocol.

### 2.0.3 Humans vs. RFID Tags

Low-cost RFID tags and other pervasive devices share many limitations with another weak computing device: human beings. We will see that in many ways, the computational capacities of people are similar to those of extremely low-cost pervasive devices.

The target cost for an EPC-type RFID tag is in the US$0.05-0.10 (5-10 cent) range. The limitations imposed at these costs in 2006 are approximated in Table 2.1. Organic printed circuits have even tighter resource constraints [111, 110].

Like people, tags can neither remember long passwords nor keep long calculations in their working memory. Tags may only be able to store a short secret of perhaps 32-128 bits, and be able to persistently store 128-512 bits overall. A working capacity of 32-128 bits of volatile memory is plausible in a low-cost tag, similar to how most human beings can maintain about seven random decimal digits in their immediate memory [87].

Neither tags nor humans can efficiently perform lengthy computations. A basic RFID tag may have a total of anywhere from 1000-10000 gates, with only 200-2000 budgeted specifically for security. (Low-cost tags achieve only the lower range of these figures.) As explained above, performing modular arithmetic over large fields or evaluating standardized cryptographic functions like AES is currently not feasible in a low-cost device nor for many human beings.

Both humans and tags must authenticate themselves to an untrusted terminal or reader in the presence of eavesdroppers. Efficiency is important in both settings. Humans will not accept a long and slow authentication process and tags must support a performance of perhaps 100 read operations per second.

Tags and people each have comparative advantages and disadvantages. Tags are

better at performing logical operations like ANDs, ORs and XORs. Tags are also better at picking random values than people – a key property of the protocols presented in this work. However, tag secrets can be completely revealed through physical attacks, such as electron microscope probing [2]. In contrast, physically attacking people tends to yield unreliable results [113].

Because of their similar sets of capabilities, this chapter adapts human authentication protocols to low-cost pervasive computing devices. The motivating human-computer authentication protocols considered were designed to allow a person to log onto an untrusted terminal while someone spies over her shoulder, without the use of any scratch paper or computational devices. Clearly, a simple password would be immediately revealed to an eavesdropper.

Such protocols are the subject of Carnegie Mellon University's HumanAut project. Earlier work by Matsumoto and Imai [80] and Matsumoto [79] propose human authentication protocols that are good for a small number of authentications [118]. Naor and Pinkas describe a human authentication scheme based on "visual cryptography" [89]. Chaum makes use of visual cryptography in a secure voter-verifiable election scheme [26]. However, this chapter focuses on the human authentication protocols of Hopper and Blum [61, 62].

## 2.1 The HB Protocol

Hopper and Blum propose a secure human authentication protocol [61, 62], which will be referred to as the HB protocol. The HB protocol is only secure against passive eavesdroppers – not active attackers. While humans may get suspicious with repeated, failed login attempts if they are actively queried by a computer, a simple tag will blindly reply to active queries. In other words, HB would not protect against skimming attacks. In Section 2.3, this work will augment the HB protocol to be secure against active adversaries that may initiate their own tag queries.

Suppose Alice and a computing device $C$ share an $k$-bit secret $\mathbf{x}$, and Alice would like to authenticate herself to $C$. Device $C$ then selects a random challenge $\mathbf{a} \in \{0,1\}^k$ and sends it to Alice. Alice computes the binary inner-product $\mathbf{a} \cdot \mathbf{x}$, then sends the result back to $C$. Finally, $C$ computes $\mathbf{a} \cdot \mathbf{x}$, and accepts the round if Alice's parity bit is correct. This protocol is illustrated in figure 2-1.

In a single round, someone imitating Alice who does not know the secret $\mathbf{x}$ will guess the correct value $\mathbf{a} \cdot \mathbf{x}$ half the time. By repeating for $r$ rounds, Alice can lower the probability of naïvely guessing the correct parity bits for all $r$ rounds to $2^{-r}$.

Of course, an eavesdropper capturing $O(k)$ valid challenge-response pairs between Alice and $C$ can quickly calculate the value of $\mathbf{x}$ through Gaussian elimination. To prevent revealing $\mathbf{x}$ to passive eavesdroppers, Alice will inject noise into her responses. Alice intentionally sends the wrong response with constant probability $\eta \in (0, \frac{1}{2})$. $C$ then authenticates (i.e. accepts as valid) Alice's identity if fewer than $\eta r$ of her responses are incorrect.

Note that there are a couple of ways to define the acceptance threshold. A reader might also accept a tag if it gets *exactly* $\eta r$ rounds incorrect. Alternatively, one may

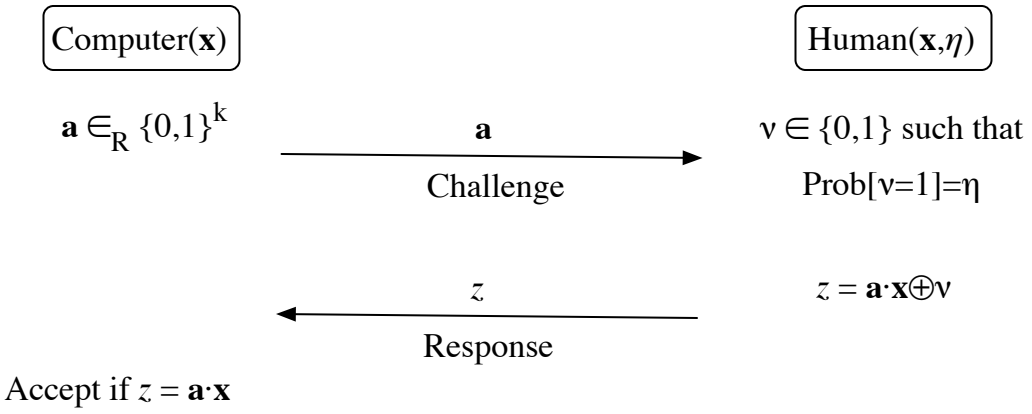


Figure 2-1: A single round of the HB authentication protocol that allows a human being to authenticate herself to a computer sharing a secret $\mathbf{x}$. The human intentionally adds noise to an $\eta$ fraction of her parity bit responses.

consider $\eta r$ to be an expected value and accept any tag that gets at most $2\eta r$ rounds incorrect.

Note that blind guessing would fail an expected $r/2$ rounds. The gap between $\eta r$ and $r/2$ will differentiate authentic parties, who know $x$, from counterfeits. For instance, if $\eta = 1/4$, then a legitimate party will get $3/4$ of the parity bits correct.

Figure 2-1 illustrates a round of the HB protocol in the RFID setting. Here, the tag plays the role of the prover (Alice) and the reader of the authenticating device $C$. Each authentication consists of $r$ rounds, where $r$ is a security parameter.

The HB protocol is very simple to implement in hardware. Computing the binary inner product $\mathbf{a} \cdot \mathbf{x}$ only requires bitwise AND and XOR operations that can be computed on the fly as each bit of $\mathbf{a}$ is received. There is no need to buffer the entire value $\mathbf{a}$.

The noise bit $\nu$ can be cheaply generated from physical properties like thermal noise, shot noise, diode breakdown noise, meta-stability, oscillation jitter, or any of a slew of other methods [7, 16, 60, 73, 97, 112]. Only a single random bit value is needed in each round. Since only a single bit of randomness is needed per round, there is less risk of being exposed to localized correlations in physical sources of randomness. This can be a problem in in chaos-based or diode breakdown random number generators, which cannot be sampled at too high of a frequency.

**Remark:** The HB protocol can be also deployed as a *privacy-preserving* identification scheme. A reader may initiate queries to a tag without actually knowing whom that tag belongs to. Based on the responses, a reader can check its database of known tag values and see if there are any likely matches. This is discussed further in Section 2.7.

## 2.2 Learning Parity in the Presence of Noise

Suppose that a passive adversary eavesdrops and captures $q$ rounds of the HB protocol over several authentications and wishes to impersonate Alice. Consider each $k$-bit challenge $\mathbf{a}$ as a row in a $q \times k$ binary matrix $\mathbf{A}$; similarly, let us view Alice's set of responses as a vector $\mathbf{z}$. Given the challenge set $\mathbf{A}$ sent to Alice, a natural attack for the adversary is to try to find a $k$-bit vector $\mathbf{x}'$ that is functionally close to Alice's secret $\mathbf{x}$. In other words, the adversary might try to compute a $\mathbf{x}'$ which, given challenge set $\mathbf{A}$ in the HB protocol, yields a set of responses that is close to $\mathbf{z}$. (Ideally, the adversary would like to figure out $\mathbf{x}$ itself.)

The goal of the adversary is akin to a problem known as the *Learning Parity in the Presence of Noise*, or LPN problem, that will be the basis of investigations in this chapter. The LPN problem involves finding a $k$-bit vector $\mathbf{x}'$ such that $|(\mathbf{A} \cdot \mathbf{x}') \oplus \mathbf{z}| \leq \eta q$, where $|\mathbf{v}|$ represents the Hamming weight of vector $\mathbf{v}$. Formally, it is as follows:

**Definition 1 (LPN Problem)** *Let $\mathbf{A}$ be a random $q \times k$ binary matrix, let $\mathbf{x}$ be a random $k$-bit vector, let $\eta \in (0, \frac{1}{2})$ be a constant noise parameter, and let $\nu$ be a random $q$-bit vector such that $|\nu| \leq \eta q$. Given $\mathbf{A}$, $\eta$, and $\mathbf{z} = (\mathbf{A} \cdot \mathbf{x}) \oplus \nu$, find a $k$-bit vector $\mathbf{x}'$ such that $|(\mathbf{A} \cdot \mathbf{x}') \oplus \mathbf{z}| \leq \eta q$.*

The LPN problem may also be formulated and referred to as as the *Minimum Disagreement Problem* [30], or the problem of finding the closest vector to a random linear error-correcting code; also known as the syndrome decoding problem [8, 76]. Syndrome decoding is the basis of the McEliece public-key cryptosystem [81] and other cryptosystems, e.g., [28, 91]. Algebraic coding theory is also central to Stern's public-key identification scheme [109]. Chabaud offers attacks that, although infeasible, help to establish practical security parameters for error-correcting-code based cryptosystems [24].

A resent result due to Regev reduces the shortest-vector problem (SVP) to learning parity with noise [100]. Several lattice-based cryptosystems, such as NTRU [58], are based on the hardness of the SVP. One caveat of the Regev reduction is that it makes use of a quantum computation in one step of the reduction. Regev conjectured that this assumption may be eliminated, but at this time it has not.

The LPN problem is known to be NP-Hard [8], and is hard even within an approximation ratio of two [56]. A long-standing open question is whether this problem is difficult for random instances. A result by Kearns proves that the LPN is not efficiently solvable in the statistical query model [70]. An earlier result by Blum, Furst, Kearns, and Lipton [10] shows that given a random $k$-bit vector $\mathbf{a}$, an adversary who could weakly predict the value $\mathbf{a} \cdot \mathbf{x}$ with advantage $\frac{1}{k^c}$ could solve the LPN problem.

The best known algorithm to solve random LPN instances is due to Blum, Kalai, and Wasserman, and has a sub-exponential, yet still non-polynomial, runtime of $2^{O(k/\log k)}$ [11]. Based on a concrete analysis of this algorithm, Section 2.8 discusses estimates for lower-bounds on key sizes for the HB and HB$^+$ protocols.

As mentioned above, the basic HB protocol is only secure against passive eavesdroppers. It is not secure against an active adversary with the ability to query tags.

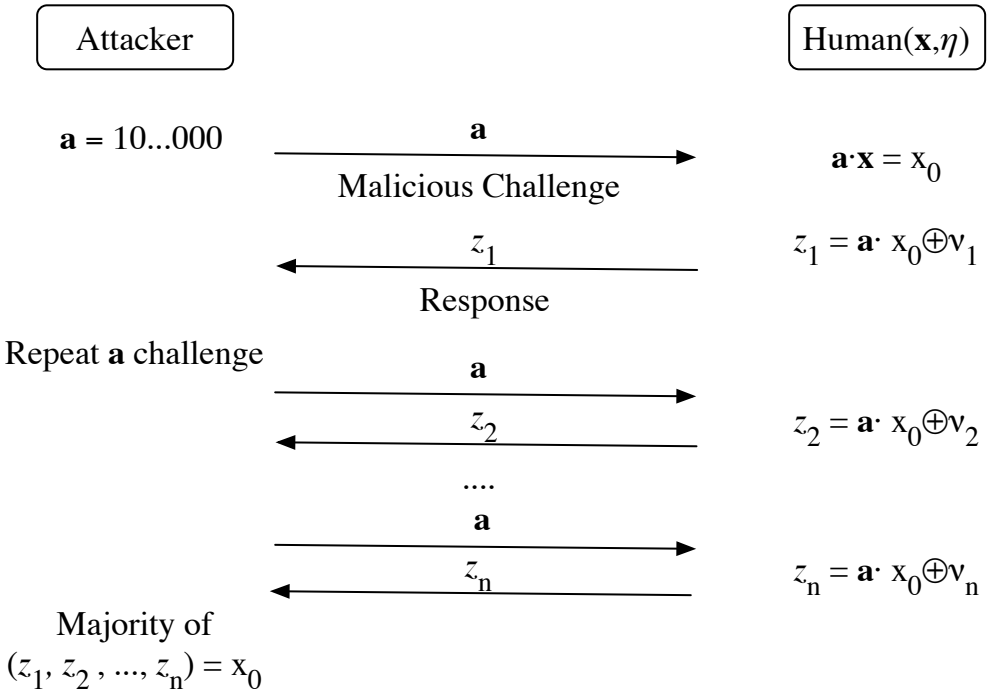


Figure 2-2: An active attacker against HB may repeat the same **a** challenge many times. In this example **a** contains a single 1 bit. The attacker may determine the corresponding bit of the secret **x** by taking the majority value of the noise parity bit samples.

To extract a secret **x** from a tag, an adversary can simply repeat the same **a** challenge multiple times. The active adversary knows that an $\eta$ fraction of the parity bit responses will be incorrect, so can simply take the majority of the responses as the true, noise-free parity bit. This attack is illustrated in figure 2-2.

The adversary can determine a noise-free parity bit by taking the majority of $q$ samples. Recall that there is a $(1-\eta)$ probability of obtaining a noise-free parity bit for a given sample, and that noise is chosen independently for each sample. By a standard Chernoff bound, the probability that at least $q/2$ samples are noise-free is $1 - exp(-q/(8(1-2\eta)^2))$. So, with high probability, if $q = O((1-2\eta)^{-2})$, then the majority value will be the true noise-free parity bit.

By collecting $O(k)$ noise-free parity bits, an active adversary can determine **x** through Gaussian elimination. To put it in practical terms, suppose that $k = 256$, that $\eta = 1/4$, and that a tag supports a conservative estimate of 50 read operations per second. An active attack could extract this tag's secret with high probability in under 2 minutes, illustrating HB's fatal weakness against skimming attacks.

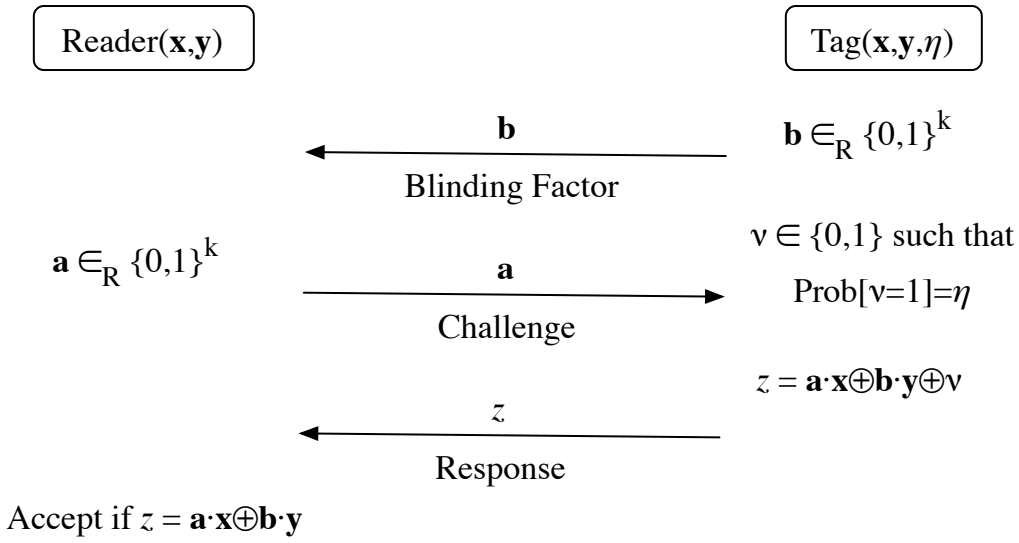


Figure 2-3: A single round of the HB$^+$ protocol

## 2.3 Authentication Against Active Adversaries

This section strengthens the HB protocol against active adversaries. The resulting protocol will be dubbed HB$^+$ and is secure against skimming attacks. HB$^+$ prevents corrupt readers from extracting tag secrets through adaptive (non-random) challenges, and thus prevents counterfeit tags from successfully authenticating themselves. Happily, HB$^+$ requires only marginally more resources than the "passive" HB protocol in the previous section.

### 2.3.1 Defending Against Active Attacks: The HB$^+$ Protocol

The HB$^+$ protocol is quite simple, and shares a familiar "commit, challenge, respond" format with classic protocols like Fiat-Shamir identification. Rather than sharing a single $k$-bit random secret $\mathbf{x}$, the tag and reader now share an additional $k$-bit random secret $\mathbf{y}$.

Unlike the case in the HB protocol, the tag in the HB$^+$ protocol first generates random $k$-bit "blinding" vector $\mathbf{b}$ and sends it to the reader. As before, the reader challenges the tag with an $k$-bit random vector $\mathbf{a}$.

The tag then computes $z = (\mathbf{a} \cdot \mathbf{x}) \oplus (\mathbf{b} \cdot \mathbf{y}) \oplus \nu$, and sends the response $z$ to the reader. The reader accepts the round if $z = (\mathbf{a} \cdot \mathbf{x}) \oplus (\mathbf{b} \cdot \mathbf{y})$. As before, the reader authenticates a tag after $r$ rounds if the tag's response is incorrect in less than $\eta r$ rounds. This protocol is illustrated in figure 2-3.

One reason that Hopper and Blum may not have originally proposed this protocol improvement is that it is inappropriate for use by humans. It requires the tag (playing the role of the human), to generate a random $k$-bit string $\mathbf{b}$ on each query. If the tag (or human) does not generate uniformly distributed $\mathbf{b}$ values, it may be possible to extract information on $\mathbf{x}$ or $\mathbf{y}$.

To convert $HB^+$ into a two-round protocol, an intuitive idea would be to have the tag transmit its $\mathbf{b}$ vector along with its response bit $z$. Being able to choose $\mathbf{b}$ *after* receiving $\mathbf{a}$, however, may give too much power to an adversarial tag. In particular, the security reduction in Section 2.4.4 relies on the tag transmitting its $\mathbf{b}$ value first. It's an open question whether there exists a secure two-round version of $HB^+$, which will be discussed further in section 2.6. Another open question is whether security is preserved if $\mathbf{a}$ and $\mathbf{b}$ are transmitted simultaneously on a duplex channel.

Beyond the requirements for the HB protocol, $HB^+$ only requires the generation of $k$ random bits for $\mathbf{b}$ and additional storage for an $k$-bit secret $\mathbf{y}$. As before, computations can be performed bitwise; there is no need for the tag to store the entire vectors $\mathbf{a}$ or $\mathbf{b}$. Overall, this protocol is still quite efficient to implement in hardware, software, or perhaps even by a human being with a decent randomness source.

### 2.3.2 Security Intuition

As explained above, an active adversary can defeat the basic HB protocol and extract $\mathbf{x}$ by making adaptive, non-random $\mathbf{a}$ challenges to the tag. In the augmented protocol $HB^+$, an adversary can also, of course, select $\mathbf{a}$ challenges. However, by selecting its own random blinding factor $\mathbf{b}$, the tag in an $HB^+$ protocol prevents an active adversary from extracting information on $\mathbf{x}$ or $\mathbf{y}$.

Since the secret $\mathbf{y}$ is independent of $\mathbf{x}$, we may think of the tag as initiating an independent, interleaved HB protocol with the roles of the participants reversed. In other words, an adversary observing $\mathbf{b}$ and $(\mathbf{b} \cdot \mathbf{y}) \oplus \nu$ should not be able to extract significant information on $\mathbf{y}$.

Recall that the value $(\mathbf{b} \cdot \mathbf{y}) \oplus \nu$ is XORed with the the output of the original, reader-initiated HB protocol, $\mathbf{a} \cdot \mathbf{x}$. This is intended to prevent an adversary from extracting information through non-random $\mathbf{a}$ challenges. Thus, the value $(\mathbf{b} \cdot \mathbf{y}) \oplus \nu$ should effectively "blind" the value $\mathbf{a} \cdot \mathbf{x}$ from both passive and active adversaries.

This observation underlies the strategy for proving the security of $HB^+$. The argument is that an adversary able to efficiently learn $\mathbf{y}$ can efficiently solve the LPN problem. In particular, an adversary that does not know $\mathbf{y}$ cannot guess $\mathbf{b} \cdot \mathbf{y}$, and therefore cannot learn information about $\mathbf{x}$ from a tag response $z$.

Blinding therefore protects against leaking the secret $\mathbf{x}$ in the face of active attacks. Without knowledge of $\mathbf{x}$ or $\mathbf{y}$, an adversary cannot create a fake tag that will respond correctly to a challenge $\mathbf{a}$. In other words, cloning will be infeasible. Section 2.4 will present a concrete reduction from the LPN problem to the security of the $HB^+$ protocol. In other words, an adversary with some significant advantage of impersonating a tag in the $HB^+$ protocol can be used to solve the LPN problem with some significant advantage.

## 2.4 Security Proofs

Section 2.4.1 presents concrete security notation used for proofs of security. Section 2.4.2 reviews key aspects of the Blum et al. proof strategy that reduces the LPN problem to the security of the HB protocol [10]. Section 2.4.3 offers a more thorough and concrete version of the Blum et al. reduction. Section 2.4.4, presents a concrete reduction from the HB protocol to the HB$^+$ protocol. Finally, in Section 2.4.5, these results are combined to show a concrete reduction of the LPN problem to the security of the HB$^+$ protocol.

### 2.4.1 Notation and Definitions

A tag-authentication system is defined in terms of a pair of probabilistic functions $(\mathcal{R}, \mathcal{T})$, namely a reader function $\mathcal{R}$ and a tag function $\mathcal{T}$. The tag function $\mathcal{T}$ is defined in terms of a noise parameter $\eta$ and a $k$-bit secret $\mathbf{x}$. In the HB$^+$ setting, $\mathcal{T}$ will include an additional $k$-bit secret $y$. Let $q$ be the maximum number of protocol invocations on $\mathcal{T}$ in the experiments described in figure 2-4.

In the HB setting, $\mathcal{T}$'s interaction with an honest reader will be captured by a set of $q$ random $k$-bit vectors $\{\mathbf{a}^{(i)}\}_{i=1}^{q}$ that for convenience is viewed as a matrix $\mathbf{A}$. Regularly, a tag would obtain $\mathbf{a}$ values adaptively from a reader. For convenience, we treat the tag as an oracle that provides flat HB transcripts to a passive eavesdropper.

For protocol HB, the fully parameterized tag function is denoted as $\mathcal{T}(x, \mathbf{A}, \eta)$. On the $i$th invocation of this protocol, $\mathcal{T}$ is presumed to output $(\mathbf{a}^{(i)}, (\mathbf{a}^{(i)} \cdot \mathbf{x}) \oplus \nu)$. Here $\nu$ is a bit of noise parameterized by $\eta$. This models a passive eavesdropper observing a round of the HB protocol. Note that the oracle $\mathcal{T}(x, \mathbf{A}, \eta)$ takes no input and essentially acts as an interface to a flat transcript.

For this protocol, the reader $\mathcal{R}_x$ takes as input a pair $(\mathbf{a}, z)$. It outputs either "accept" or "reject" if it believes the tag is authentic or not, as defined by the HB and HB$^+$ protocols. That is, the reader will accept a tag as authentic if it passes at least $(1 - \eta) \cdot q$ rounds correctly.

For protocol HB$^+$, the fully parameterized tag function is denoted as $\mathcal{T}(x, y, \eta)$. This oracle internally generates random blinding vectors $\mathbf{b}$. On the $i$th invocation of $\mathcal{T}$ in the protocol, the tag outputs some random $\mathbf{b}^{(i)}$, takes a challenge vector $\mathbf{a}^{(i)}$ (that could depend on $\mathbf{b}^{(i)}$) as input, and outputs $z = (\mathbf{a}^{(i)} \cdot \mathbf{x}) \oplus (\mathbf{b}^{(i)} \cdot \mathbf{y}) \oplus \nu$. This models an active adversary querying a tag in a round of the HB$^+$ protocol. For this protocol, the reader $\mathcal{R}_{x,y}$ takes as input a triple $(\mathbf{a}, \mathbf{b}, z)$ and outputs either "accept" or "reject".

A two-phase attack model involving an adversary comprising a pair of functions $\mathcal{A} = (\mathcal{A}_{query}, \mathcal{A}_{clone})$, a reader $\mathcal{R}$, and a tag $\mathcal{T}$ will be used for both protocols HB and HB$^+$. In the first, "query" phase, the adversarial function $\mathcal{A}_{query}$ has oracle access to $\mathcal{T}$ and outputs some state $\sigma$.

The second, "cloning" phase involves the adversarial function $\mathcal{A}_{clone}$. The function $\mathcal{A}_{clone}$ takes as input a state value $\sigma$. In HB$^+$, it outputs a blinding factor $\mathbf{b}'$ (when given the input command "initiate"). In both HB and HB$^+$, when given the

Experiment $\mathbf{Exp}^{HB}_{\mathcal{A},D}[k,\eta,q]$
$x \overset{R}{\leftarrow} \{0,1\}^k$;
$\mathbf{A} \overset{R}{\leftarrow} D$
$\sigma \leftarrow \mathcal{A}(k,\eta,q)^{\mathcal{T}(x,\mathbf{A},\eta)}_{query}$;
$\mathbf{a}' \overset{R}{\leftarrow} \{0,1\}^k$;
$z' \leftarrow \mathcal{A}_{clone}(\sigma, \mathbf{a}',$ "guess");
Output $\mathcal{R}_x(\mathbf{a}', z')$.

Experiment $\mathbf{Exp}^{HB^+}_{\mathcal{A}}[k,\eta,q]$
$x, y \overset{R}{\leftarrow} \{0,1\}^k$;
$\sigma \leftarrow \mathcal{A}(k,\eta,q)^{\mathcal{T}(x,y,\eta)}_{query}$;
$\mathbf{b}' \leftarrow \mathcal{A}_{clone}(\sigma,$ "initiate");
$\mathbf{a}' \overset{R}{\leftarrow} \{0,1\}^k$;
$z' \leftarrow \mathcal{A}_{clone}(\sigma, \mathbf{a}', \mathbf{b}',$ "guess");
Output $\mathcal{R}_{x,y}(\mathbf{a}', \mathbf{b}', z')$.

Figure 2-4: HB and HB$^+$ attack experiments with concrete parameters

input command "guess", $\mathcal{A}_{clone}$ takes the full experimental state as input, and outputs a response bit $z'$.

A protocol invocation is presumed to take some fixed amount of computation steps or runtime (as would be the case, for example, in an RFID system). The total protocol time is characterized by three parameters: the number $q$ of queries to a $\mathcal{T}$ oracle; the computational runtime $t_1$ of $\mathcal{A}_{query}$; and the computational runtime $t_2$ of $\mathcal{A}_{clone}$. Let $D$ be some distribution of $q \times k$ matrices. Let $\overset{R}{\leftarrow}$ denote sampling a random value from a given distribution. Other notation should be clear from context. $\mathbf{Exp}^{HB}$ and $\mathbf{Exp}^{HB^+}$ are illustrated in figure 2-4.

Consider $\mathcal{A}$'s advantage for key-length $k$, noise parameter $\eta$, over $q$ rounds. Again, $\mathcal{R}$ will follow either the HB or HB$^+$ protocol and only accept a tag as authentic if an expect $(1-\eta)q$ rounds are correct. In the case of the HB-attack experiment, this advantage will be over matrices $\mathbf{A}$ drawn from the distribution $D$:

$$\mathsf{Adv}^{HB}_{\mathcal{A},D}(k,\eta,q) = \left| Pr\left[\mathbf{Exp}^{HB}_{\mathcal{A},D}[k,\eta,q] = \text{"accept"}\right] - \frac{1}{2} \right|$$

Let $Time(t_1,t_2)$ represent the set of all adversaries $\mathcal{A}$ with runtimes $t_1$ and $t_2$, respectively. Denote the maximum advantage over $Time(t_1,t_2)$:

$$\mathsf{Adv}^{HB}_{D}(k,\eta,q,t_1,t_2) = \max_{\mathcal{A} \in Time(t_1,t_2)} \{\mathsf{Adv}^{HB}_{\mathcal{A},D}(k,\eta,q)\}$$

The definitions for $\mathsf{Adv}$ are exactly analogous for HB$^+$-attack, except that there is no input distribution $D$, as adversarial queries are active.

### 2.4.2 Blum et al. Proof Strategy Outline

Given an adversary $\mathcal{A}$ that achieves the advantage $\mathsf{Adv}^{HB}_{\mathcal{A},U}(k,q,\eta,t_1,t_2) = \epsilon$, where $U$ is the uniform distribution of $q \times k$ binary matrices, Blum et al. [10] offer a proof strategy to extract bits of $x$, and thus solve the LPN problem. If $\epsilon$ is greater than $1/\text{poly}(k)$ for some poly$(\cdot)$, then $x$ can be extracted by their reduction in polynomial time.

To extract the $i$th bit of the secret $x$, the Blum et al. reduction takes a given LPN instance $(\mathbf{A}, \mathbf{z})$ and randomly modifies it to produce a new instance $(\mathbf{A}', \mathbf{z}')$.

The modification involves two steps. First, a vector $\mathbf{x}'$ is chosen uniformly at random and $\mathbf{z}' = (\mathbf{z} \oplus \mathbf{A}) \cdot \mathbf{x}' = (\mathbf{A} \cdot (\mathbf{x} \oplus \mathbf{x}')) \oplus \nu$ is computed. Note that thanks to the random selection of $\mathbf{x}'$, the vector $(\mathbf{x} \oplus \mathbf{x}')$ is uniformly distributed. Second, the $i$th column of $\mathbf{A}$ is replaced with random bits.

To view this another way, denote the subspace of matrices obtained by uniformly randomizing the $i$th column of $\mathbf{A}$ as $R_i^{\mathbf{A}}$. The second step of the modification involves setting $\mathbf{A}' \stackrel{R}{\leftarrow} R_i^{\mathbf{A}}$. Once computed as described, the modified problem instance $(\mathbf{A}', \mathbf{z}')$ is fed to an HB adversary $\mathcal{A}_{query}$.

Suppose that the $i$th bit of $(\mathbf{x} \oplus \mathbf{x}')$, denoted $(x \oplus x')_i$, is a binary '1'. In this case, since $\mathbf{A}$ is a randomly distributed matrix (because HB challenges are random), and the secret $\mathbf{x}$ is also randomly distributed, the bits of $\mathbf{z}'$ are random. In other words, thanks to the '1' bit, the randomized $i$th row of $\mathbf{A}'$ "counts" in the computation of $\mathbf{z}'$, which therefore comes out random. Hence $\mathbf{z}'$ contains no information about the correct value of $\mathbf{A} \cdot (\mathbf{x} \oplus \mathbf{x}')$ or about the secret $x$. Since $\mathcal{A}_{query}$ cannot pass any meaningful information in $\sigma$ to $\mathcal{A}_{clone}$ in this case, $\mathcal{A}_{clone}$ can do no better than random guessing of parity bits, and enjoys no advantage.

In contrast, suppose that $(x \oplus x')_i$, is a binary '0'. In this case, the $i$th row of $\mathbf{A}'$ does not "count" in the computation of $\mathbf{z}'$, and does not have a randomizing effect. Hence $\mathbf{z}'$ may contain meaningful information about the secret $\mathbf{x}$ in this case. As a result, when $\mathcal{A}_{clone}$ shows an advantage over modified problem instances $(\mathbf{A}', \mathbf{z}')$ for a particular fixed choice of $\mathbf{x}'$, it is clear for those instances that $(x \oplus x')_i = 0$, i.e. $x_i = x'_i$.

In summary then, the Blum et al. reduction involves presentation of suitably modified problem instances $(\mathbf{A}', \mathbf{z}')$ to HB adversary $\mathcal{A}$. By noting choices of $\mathbf{x}'$ for which $\mathcal{A}$ demonstrates an advantage, it is possible in principle to learn individual bits of the secret $\mathbf{x}$. With presentation of enough modified problem instances to $\mathcal{A}$, it is possible to learn $\mathbf{x}$ completely with high probability.

### 2.4.3 Reduction from LPN to HB-attack

This section shows a concrete reduction from the LPN problem to the HB-attack experiment. This is essentially a concrete version of Blum et al.'s asymptotic reduction strategy from [10] and is an important step in proving Theorem 1, which is a main result of this chapter.

Unfortunately, the original Blum et al. proof strategy does not account for the fact that while $\mathcal{A}$'s advantage may be non-negligible over random matrices, it may actually be negligible over modified $(\mathbf{A}', \mathbf{z}')$ values, i.e., over the distribution $R_i^{\mathbf{A}}$. Matrices are not independent over this distribution: Any two sample matrices are identical in all but one column. Thus, it is possible in principle that $\mathcal{A}$ loses its advantage over this distribution of matrices and that the reduction fails to work. This problem is remedied here.

We address the problem by modifying a given sample matrix only once. A modified matrix $\mathbf{A}'$ in this reduction is uniformly distributed. This is because it is chosen uniformly from a random $R_i^{\mathbf{A}}$ subspace associated with a random matrix $\mathbf{A}$. Addition-

ally, since a fresh sample is used for each trial, the modified matrices are necessarily independent of each other. The trade-off is that $kL$ times as many sample matrices are needed for this reduction, where $L$ is the number of trials per bit.

This is an inefficient solution in terms of samples. It is entirely possible that the adversary's advantage is preserved when, for each column $j$, samples are drawn from the $R_i^{\mathbf{A}_j}$ subspace for a matrix $\mathbf{A}_j$. It might even be possible to devise a rigorous reduction that uses a single matrix $\mathbf{A}$ for all columns. These are left as open questions.

**Lemma 1** *Let $\mathsf{Adv}_U^{HB}(k, \eta, q, t_1, t_2) = \epsilon$, where $U$ is a uniform distribution over binary matrices $\mathbb{Z}_2^{q \times k}$, and let $\mathcal{A}$ be an adversary that achieves this $\epsilon$-advantage. Then there is an algorithm $\mathcal{A}'$ with running time $t_1' \leq kLt_1$ and $t_2' \leq kLt_2$, where $L = \frac{8(\ln k - \ln\ln k)}{(1-2\eta)^2}(\frac{1+\epsilon}{\epsilon})^2$, that makes $q' \leq kLq + 1$ queries that can correctly extract all $k$ bits of $x$ with probability $\epsilon' \geq \frac{1}{k}$.*

**Proof:** Given an adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A},U}^{HB}(k, q, \eta, t_1, t_2) = \epsilon$, this proof will show how to construct a simulator $\mathcal{S}$ to extract all bits of an HB secret $\mathbf{x}$ with high probability. Thus, $\mathcal{S}$ will be able solve the LPN problem.

Consider a particular LPN instance $(\widehat{\mathbf{A}}, \widehat{\mathbf{z}})$ that is given as input. The goal of $\mathcal{S}$ will be to use $\mathsf{Adv}$ and extract the LPN secret $\mathbf{x}$. The simulator $\mathcal{S}$ will first select one sample row from $(\widehat{\mathbf{A}}, \widehat{\mathbf{z}})$ at random and denote it $(\widehat{\mathbf{a}}, \widehat{z})$. $\mathcal{S}$ splits the remaining samples into $kL$ sets of size $q$ ($L$ will be defined later). The simulator will then replace a random $i$th column of $L$ different samples with random bits and randomize the associated $\mathbf{z}$ value as described in Section 2.4.2. Denote these samples as $(\mathbf{A}', \mathbf{z}')$, respectively.

$\mathcal{S}$ will then input each $(\mathbf{A}', \mathbf{z}')$ sample to $\mathcal{A}_{query}$. In the cloning phase, $\mathcal{S}$ replaces the $i$th bit of $\widehat{\mathbf{a}}$ with a random bit and challenges $\mathcal{A}_{clone}$ for the result. The simulator $\mathcal{S}$ knows the noisy sample $\widehat{z}$, thus can verify whether $\mathcal{A}_{clone}$'s result matches.

Recall from Section 2.4.2 that $\mathbf{z}'$ was generated by choosing a random $\mathbf{x}'$ and adding the value $\mathbf{A}' \cdot \mathbf{x}'$ to $\mathbf{z}$. If $(x \oplus x')_i = 0$, then replacing the $i$th column of $\mathbf{A}'$ does not affect $\mathbf{z}'$ and $(\mathbf{A}', \mathbf{z}')$ is a valid LPN instance. The hope is that $\mathcal{A}$ would maintain its $\epsilon$-advantage over this distribution of samples. However, it is conceivable that the adversary $\mathcal{A}$'s advantage over this modified distribution of samples whose $i$th secret bit is necessarily zero is less than $\epsilon$

However, since the samples are drawn from a valid LPN distribution, the adversary must still maintain an $\epsilon$-advantage over all secrets. Thus, any under-performance over the distribution of '0'-valued $i$th bits is made up over the distribution of '1'-valued $i$th bits.

Denote the event that the $i$th secret bit is zero as $Z$, and when it is one as $\bar{Z}$. Suppose $Pr[\mathcal{A} \text{ succeeds} \mid Z] = (\epsilon - \delta)$ and $Pr[\mathcal{A} \text{ succeeds} \mid \bar{Z}] = (\epsilon + \delta)$. If $\delta$ is significantly large enough, then $\mathcal{S}$ can simply run $\mathcal{A}$ on the original, unaltered samples $\widehat{\mathbf{A}}$ and observe its performance. An adversary that achieves advantage less than $\epsilon$ would indicate $Z$, while advantage greater than $\epsilon$ would indicate $\bar{Z}$. Note that a significant $\delta$ can be detected by generating random LPN instances and measuring $\mathcal{A}$'s performance conditioned on $Z$ and $\bar{Z}$ events.

Using this naïve approach, $\mathcal{A}$ correctly outputs the $i$th bit when either both $\widehat{z}$ and $\mathcal{A}$ are correct, or they are both wrong (recall that $\widehat{z}$ is a noisy sample). The event that both $\widehat{z}$ and $\mathcal{A}$ are correct, or both are wrong occurs with probability $(1-\eta)(\frac{1}{2}+\frac{\delta}{2\epsilon})+\eta(\frac{1}{2}-\frac{\delta}{2\epsilon})$. Thus $\mathcal{A}$ would guess $x_i$ with expected probability $\frac{1}{2}+\frac{1}{2}(1-2\eta)\frac{\delta}{\epsilon}$.

However, if we we use the modified $\mathbf{A}'$ samples, the $\mathcal{A}$ will correctly guess $x_i$ with expected probability $\frac{1}{2}+\frac{1}{2}(1-2\eta)(\epsilon-\delta)$. Thus, if $\frac{\delta}{\epsilon} \geq (\epsilon-\delta)$, it would be advantageous for $\mathcal{S}$ to use the naïve method. Simplifying, it will make sense to use the naïve method if $\delta \geq \frac{\epsilon^2}{1+\epsilon}$.

If $\delta \geq \frac{\epsilon^2}{1+\epsilon}$, then there exists a simulator that simply permutes columns of the original challenges $\widehat{\mathbf{A}}$ and maintains alignment with the corresponding rows of $\widehat{\mathbf{z}}$. Then a simulator can run the naïve attack to determine each key bit with advantage $\frac{1}{2}+\frac{1}{2}(1-2\eta)(\frac{\delta}{\epsilon})$ per trial. The simulator can permute columns such that each key bit is assigned to the $i$th column exactly $L$ times. The existence of this simulator relies on the assumption that the original samples are drawn from a random distribution, which is closed under permutation.

Thus, in the worst case we have that $\delta = \frac{\epsilon^2}{1+\epsilon}$ and $(\epsilon-\delta) = \frac{\epsilon}{1+\epsilon}$. For convenience, denote $\mathcal{A}$'s advantage at guessing a bit per trial as $\widehat{\epsilon} = \frac{1}{2}(1-2\eta)(\frac{\epsilon}{1+\epsilon})$.

Consider repeating $L$ randomly modified trials per $k$ bits of $\mathbf{x}$ and taking the majority of the outcome for each bit. By a Chernoff bound, after $L$ trials each guessed bit will be correct with probability $p = (1-exp(\frac{-L\widehat{\epsilon}^2}{1+2\widehat{\epsilon}})) \leq (1-exp(\frac{-L\widehat{\epsilon}^2}{2}))$. Thus, all $k$ bits will be correct with probability $p^k$.

Thus, if $\mathcal{A}$ requires $q$ samples and runs in $(t_1, t_2)$ time, this reduction will extract all bits of $x$ using $q' = kLq+1$ samples and running in time $t' = tLk$ with probability:

$$(1-e^{\frac{-L\widehat{\epsilon}^2}{2}})^k \approx exp\left(\frac{-k}{exp(\frac{L\widehat{\epsilon}^2}{2})}\right)$$

Let $L = \frac{2(\ln k - \ln\ln k)}{\widehat{\epsilon}^2}$. Plugging this into the above formula gives us a success probability of $\frac{1}{k}$. Substituting in for $\widehat{\epsilon}$, we get that $L = 8(\ln k - \ln\ln k)\left((1+\epsilon)/((1-2\eta)\epsilon)\right)^2$. Thus, we can express $t'_1 = t_1 Lk$ and $t'_2 = t_2 Lk$ concretely in terms of $k$ and $\eta$. □

### 2.4.4 Reduction from HB-attack to HB$^+$-attack

The next lemma is the main result of this chapter, namely a reduction of the security of the $HB^+$ protocol to the security of the $HB$ protocol. This lemma implies that an HB$^+$-attack adversary with $\zeta$-advantage can be used to build an HB-attack adversary with advantage $\frac{\zeta^3(k-2)-2}{4k}$. Concrete costs of this reduction will be used for Theorem 1.

We note that the lemma is only meaningful for relatively large advantage values $\zeta$. Small advantages, however, can be boosted through the standard technique of taking majority output from multiple (polynomial) adversarial executions.

**Lemma 2** *If* $\mathsf{Adv}_U^{HB^+}(k,\eta,q,t_1,t_2) = \zeta$, *then*

$$\mathsf{Adv}_U^{HB}(k,\eta,q',t'_1,t'_2) \geq \frac{\zeta^3(k-2)-2}{4k}$$

*where* $q' \leq q(2 + \log_2 q)$, $t'_1 \leq kq't_1$, $t'_2 \leq 2kt_2$, *and* $k \geq 9$.

Lemma 2 is the main technical core of this chapter. It is worth briefly explaining the proof intuition. The proof naturally involves a simulation where the HB-attack adversary $\mathcal{A}$ makes calls to the furnished HB$^+$-attack adversary, which we call $\mathcal{A}^+$. In other words, $\mathcal{A}$ simulates the environment for $\mathbf{Exp}^{HB^+}_{\mathcal{A}^+}$. The goal of $\mathcal{A}$ is to use $\mathcal{A}^+$ to compute a correct target response $w$ to an HB challenge vector $\mathbf{a}$ that $\mathcal{A}$ itself receives in an experiment $\mathbf{Exp}^{HB}_{\mathcal{A}}$.

$\mathcal{A}$ makes its calls to $\mathcal{A}^+$ in a special way: It "cooks" transcripts obtained from its own HB oracle before passing them to $\mathcal{A}^+$ during its simulation of the query phase of $\mathbf{Exp}^{HB^+}$. The "cooked" transcripts are such that the target value $w$ is embedded implicitly in a secret bit of the simulated HB$^+$ oracle.

In its simulation of the cloning phase of $\mathbf{Exp}^{HB^+}$, the adversary $\mathcal{A}$ extracts the embedded secret bit using a standard cryptographic trick. After $\mathcal{A}^+$ has committed a blinding value $\mathbf{b}$, $\mathcal{A}$ rewinds $\mathcal{A}^+$ to as to make two different challenges $\mathbf{a}^{(0)}$ and $\mathbf{a}^{(1)}$ relative to $\mathbf{b}$. By looking at the difference in the responses, $\mathcal{A}$ can extract the embedded secret bit and compute its own target response $w$.

There are two main technical challenges in the proof. The first is finding the right embedding of $w$ in a secret bit of the simulated HB$^+$-oracle. Indeed, this approach is somewhat surprising. One might intuitively expect $\mathcal{A}$ instead to cause $\mathcal{A}^+$ to emit a *response* equal to $w$ during the simulation; after all, $w$ itself is intended to be a tag response furnished by $\mathcal{A}$, rather than a secret bit. (We could not determine a good way to have $w$ returned as a response.) The second challenge comes in the rewinding and extraction. There is the possibility of a non-uniformity in the responses of $\mathcal{A}^+$.

**Proof:** Suppose there exists an HB$^+$ adversary $\mathcal{A}^+$ with advantage $\mathsf{Adv}^{HB^+}_{\mathcal{A}^+,U}(k, \eta, q, t_1, t_2) = \zeta$, where $\zeta$ is non-negligible in $k$. This adversary will be used to construct an HB adversary $\mathcal{A}$. As an HB adversary, $\mathcal{A}$ queries a tag oracle $\mathcal{T}(x, \mathbf{A}, \eta)$ in the query phase of $\mathbf{Exp}^{HB}_{\mathcal{A}}$. We denote the challenge-response pairs from this phase by $(\mathbf{A}, w) = \{\mathbf{a}^{(i)}, w^{(i)}\}_{i=1}^{q+uq}$. (Note that we assume a "pool" here of $uq$ extra challenge-response pairs, where $u$ is a multiplier to be defined later.) In the cloning phase of $\mathbf{Exp}^{HB}_{\mathcal{A}}$, $\mathcal{A}$ takes a challenge vector $\mathbf{a}$ and aims to output a correct response $w = \mathbf{a} \cdot \mathbf{x}$. To accomplish this goal and determine the target value $w$, $\mathcal{A}$ makes specially formulated calls to the adversary $\mathcal{A}^+$ between its experimental phases, as we now explain.

In its calls to $\mathcal{A}^+$ during the simulated query phase, the adversary $\mathcal{A}$ simulates responses for an HB$^+$ tag oracle $\mathcal{T}(x^+, y^+, \eta)$. To do so, it takes responses from its own tag oracle $\mathcal{T}(x, \mathbf{A}, \eta)$ and "folds in" its own $k$-bit secret $\mathbf{s}$ before passing them to $\mathcal{A}^+$. In effect, $\mathcal{A}$ uses $\mathbf{s}$ as the tag oracle secret $\mathbf{x}^+$: Since the challenges $\{\mathbf{a}^{+(i)}\}$ that are XORed with $\mathbf{x}^+$ are selected actively by $\mathcal{A}^+$, the adversary $\mathcal{A}$ must have knowledge of $\mathbf{x}^+$ in order to perform the simulation successfully. In contrast, $\mathcal{A}$ itself chooses the blinding factors $\{\mathbf{b}^{+(i)}\}$ that are XORed with $\mathbf{y}^+$ in the query phase. Therefore, $\mathcal{A}$ is able to perform its simulation with $\mathbf{y}^+ = \mathbf{x}$, i.e., since it controls the challenges, it can incorporate the data $(\mathbf{A}, w)$ here that it harvested (passively) in $\mathbf{Exp}^{HB}_{\mathcal{A}}$.

Let $s[i]$ denote the $i$th bit of $\mathbf{s}$. The adversary $\mathcal{A}$ selects all bits of the secret $\mathbf{s}$ at random *except* $s[j]$. It reserves the bit $s[j]$ as a special unknown one; in its simulation with $\mathcal{A}^+$, it implicitly embeds the target value $w$ in $s[j]$, as we shall explain.

Let us now describe how $\mathcal{A}$ executes the query and cloning phases for $\mathcal{A}^+$ in its simulation of $\mathbf{Exp}_{\mathcal{A}^+}^{HB^+}$.

**Query phase:** Recall that in this phase, $\mathcal{A}_{query}^+$ queries an $HB^+$ tag oracle $\mathcal{T}(x^+, y^+, \eta)$. Let us consider the $m$th query made by $\mathcal{A}_{query}^+$, which we denote by $\mathbf{a}^{+(m)}$. Before the query is made, $\mathcal{A}$ selects a random bit $g^{(m)}$. This is $\mathcal{A}$'s guess at the query bit $a^{+(m)}[j]$. If $g^{(m)} = 0$, then $\mathcal{A}$ sets $\mathbf{b}^{+(m)} = \mathbf{a}^{(m)}$. If $g^{(m)} = 1$, it sets $\mathbf{b}^{+(m)} = \mathbf{a}^{(m)} \oplus \mathbf{a}$. $\mathcal{A}$ passes the blinding factor $\mathbf{b}^{+(m)}$ to $\mathcal{A}^+$ as the first protocol flow.

If $\mathcal{A}$'s guess $g^{(m)}$ is incorrect, i.e., $g^{(m)} \neq a^{+(m)}[j]$, then $\mathcal{A}$ rewinds to the beginning of the $m$th query. It discards the pair $(\mathbf{a}^{(m)}, w^{(m)})$ from $(\mathbf{A}, w)$ and replaces it with the next challenge-response pair. In effect, $\mathcal{A}$ draws from the "pool" of extra challenge-response pairs in $(\mathbf{A}, w)$. It halts and outputs a random guess at $w$ if the "pool" is exhausted. $\mathcal{A}$ then repeats its simulation for the $m$th query with a new guess $g^{(i)}$ and the new challenge-response pair.

If $\mathcal{A}$'s guess $g^{(i)}$ is correct, then $\mathcal{A}$ computes its response bit as:

$$z^{+(m)} = \bigoplus_{i \neq j} (a^{+(m)}[i] \cdot s[i]) \oplus w^{(m)}$$

If $g^{(m)} = a^{+(m)}[j] = 0$, then observe that there is an omitted term $u = a^{+(m)}[j]s[j]$ in this response bit; since $a^{+(m)}[j] = 0$, this omitted value $u = 0$, so the response $z^{+(m)}$ is still correct. If $g^{(m)} = a^{+(m)}[j] = 1$, then the omitted term $u = a^{+(m)}[j]s[j] \oplus w = s[j] \oplus w$. In other words, the response is correct if and only if $s[j] = w$. This is how $\mathcal{A}$ embeds the target value $w$ in the secret bit $s[j]$ (without knowing $w$). $\mathcal{A}$ then adds noise to its response according to probability value $\eta$ before transmitting it to $\mathcal{A}^+$.

**Cloning phase:** In the cloning phase, the goal of $\mathcal{A}$ is to extract the target value $s[j] = w$ from $\mathcal{A}^+$. In this phase, recall that $\mathcal{A}_{clone}^+$ attempts to simulate the oracle $\mathcal{T}(x^+, y^+, \eta)$. Thus, $\mathcal{A}^+$ first outputs a blinding factor, which we denote by $\hat{\mathbf{b}}$; then $\mathcal{A}$ provides a challenge value $\hat{\mathbf{a}}$. Finally, $\mathcal{A}^+$ outputs a response bit $\hat{z}$. If correct, the value $\hat{z} = (\hat{\mathbf{a}} \cdot \mathbf{x}^+) \oplus (\hat{\mathbf{b}} \cdot \mathbf{y}^+) = (\hat{\mathbf{a}} \cdot \mathbf{s}) \oplus (\hat{\mathbf{b}} \cdot \mathbf{x})$.

$\mathcal{A}$ selects a random pair of challenge values $(\hat{\mathbf{a}}^{\mathbf{0}}, \hat{\mathbf{a}}^{\mathbf{1}})$. It selects these such that they differ in the $j$th bit. Assume without loss of generality that $\hat{a}^{(0)}[j] = 0$ and $\hat{a}^{(1)}[j] = 1$. $\mathcal{A}$ then initiates an interaction with $\mathcal{A}^+$. It receives the blinding factor $\hat{\mathbf{b}}$. It then transmits challenge $\hat{\mathbf{a}}^{(0)}$, receiving response bit $\hat{z}^{(0)}$ from $\mathcal{A}^+$. It rewinds $\mathcal{A}^+$ and likewise transmits challenge $\hat{\mathbf{a}}^{(1)}$ to get response bit $\hat{z}^{(1)}$.

Suppose that both $\hat{z}^{(0)}$ and $\hat{z}^{(1)}$ are correct. Then:

$$\hat{z}^{(0)} \oplus \hat{z}^{(1)} = (\hat{\mathbf{a}}^{(0)} \cdot \mathbf{s}) \oplus (\hat{\mathbf{a}}^{(1)} \cdot \mathbf{s}) = \left( \sum_{i \neq j} (\hat{a}^{(0)}[i] \oplus \hat{a}^{(1)}[i]) \cdot s[i] \right) \oplus s[j]$$

Since $\mathcal{A}$ knows all bits of $s$ except $s[j]$, it can compute the first term here, and thus the target value $w = s[j]$. If both responses $\hat{\mathbf{z}}^{(0)}$ and $\hat{\mathbf{z}}^{(1)}$ are *incorrect*, the same computation works: The errors will cancel out.

What is the probability, given that $\hat{\mathbf{a}}^{(0)}$ and $\hat{\mathbf{a}}^{(1)}$ differ in the $j$th bit, that they yield like responses? To answer this question, it is necessary to determine the probability that $\hat{z}^{(0)}$ and $\hat{z}^{(1)}$ are simultaneously either both correct or both incorrect. Let $Z_0$ and $Z_1$ be random variables, where for $d \in \{0, 1\}$, $Z_d = 1$ if $\hat{z}^{(d)}$ is correct, and vice versa. It will be necessary to compute $\mathsf{Pr}[Z_0 = Z_1]$. Since the adversary has no way of knowing $j$ in the course of the simulation, assume in computing this probability that $j$ is selected *a posteriori*, i.e., after the cloning phase. It is important to note, however, that $Z_0$ and $Z_1$ are not identically distributed. In particular, the responses of the adversary $\mathcal{A}^+$ are *conditioned on the fact that* $\hat{\mathbf{a}}^{(0)}$ *and* $\hat{\mathbf{a}}^{(1)}$ *differ in a single bit.*

For this reason, it is necessary to first prove a technical lemma, which bounds the effect of this conditioning. Lemma 3 bounds the ability of an adversary to cause failures in the simulation in Lemma 2 in a step where we provide challenge vectors that differ in a random bit position. Here we let $v[j]$ denote the $j$th bit of a vector $v$, and let $\in_R$ denote uniform random selection from a set:

**Lemma 3** *Consider an experiment that takes as input a matrix* $\mathbf{A}$ *and a* $k$*-bit vector, where* $k \geq 9$*. The experiment yields either a '0' or a '1' as output. Let* $p_{\mathbf{A}}$ *denote the probability of a '1' output over random vectors of* $k$ *bits for a matrix* $\mathbf{A}$*. Suppose a pair of random* $k$*-bit vectors* $v_0$ *and* $v_1$ *is selected such that* $v_0[j] = 0$ *and* $v_1[j] = 1$ *for random* $j \in \{1, \ldots, k\}$*. Let* $p'_{\mathbf{A}}$ *be the probability that for vectors thus selected, both yield a '0' or both vectors yield a '1'. If* $p_{\mathbf{A}} \geq 1/2 + \epsilon$*, then* $p'_{\mathbf{A}} \geq 1/2 + \epsilon'$ *for* $\epsilon' = \epsilon^3/2 - (\epsilon^3 + 1)/k$*.*

**Proof:** Suppose that $v_0$ and $v_1$ are selected as in the statement of the Lemma 3, i.e., with a '0' and '1' fixed respectively at a random position $j$. Observe that for a set $S$ of $k$-bit vectors such that $|S| = 2^{k-d}$, $\mathsf{Pr}[v_0, v_1 \in S]$ is minimized when $S$ consists of vectors whose first $d$ bits are equal. Consequently, for $|S| > 2^{k-d}$, it is the case that:

$$\mathsf{Pr}[u_0[j] = u_1[j] \,|\, u_0, u_1 \in_R S, j \in_R \{1, \ldots, k\}] \geq (k - d)/k. \tag{2.1}$$

For a particular matrix $\mathbf{A}$, there is a set $S_{\mathbf{A}}$ of vectors for which the experiment outputs '1', where $|S_{\mathbf{A}}| = p_{\mathbf{A}} 2^k$. For clarity, we drop the subscript and denote this set by $S$. Let $\overline{S}$ denote the complementary set. We shall also assume $j \in_R \{1, \ldots, k\}$ as appropriate in what follows.

By Bayes's rule and eq. 2.1, we have $\mathsf{Pr}[v_0, v_1 \in S \,|\, v_0[j] = 0, v_1[j] = 1] = \mathsf{Pr}[(v_0, v_1 \in S) \bigwedge (v_0[j] = 0, v_1[j] = 1)]/\mathsf{Pr}[v_0[j] = 0, v_1[j] = 1] \geq p_{\mathbf{A}}^2(1 - \lceil \log_2 p_{\mathbf{A}} \rceil / k)$. Now consider two cases for a particular matrix $\mathbf{A}$.

**Case 1,** $p_{\mathbf{A}} \leq 1/4$**:** In this case, $p'_{\mathbf{A}} \geq \mathsf{Pr}[v_0, v_1 \in \overline{S} \,|\, v_0[j] = 0, v_1[j] = 1] > (3/4)^2(k - 1/k)$. As $k \geq 9$, it follows that $p'_{\mathbf{A}} \geq 1/2$.

**Case 2,** $p_{\mathbf{A}} > 1/4$**:** Here, $p'_{\mathbf{A}} = \mathsf{Pr}[v_0, v_1 \in S \,|\, v_0[j] = 0, v_1[j] = 1] + \mathsf{Pr}[v_0, v_1 \in \overline{S} \,|\, v_0[j] = 0, v_1[j] = 1] > p_A^2 + (1 - p_A)^2(k - 2/k) = (2p_A^2 - 2p_A + 1)(k - 2/k)$. Note that this last term is minimized for $p_{\mathbf{A}} = 1/2$, in which case $p'_{\mathbf{A}} = (k - 2)/2k$.

Since $p = 1/2 + \epsilon$, it is straightforward to show that $p_{\mathbf{A}} \geq 1/2 + \epsilon/2$ for greater than an $\epsilon$-fraction of matrices $\mathbf{A}$. For such matrices, $p'_{\mathbf{A}} > (2p_A^2 - 2p_A + 1)(k - 2/k) = (1/2 + \epsilon^2/2)(\frac{k-2}{k})$. Thus, $p' = E[p'_{\mathbf{A}}] > (1-\epsilon)(\frac{k-2}{2k}) + \epsilon(1/2 + \epsilon^2/2)(\frac{k-2}{2k}) = (\frac{\epsilon^3+1}{2})(\frac{k-2}{k})$. The lemma then follows by straightforward algebraic manipulation. (End of Lemma 3 proof.) □

Recall that $\mathsf{Adv}^{HB}_{\mathcal{A}^+}(k, \eta, q, t_1, t_2) = \zeta$ by assumption; thus, the $p = 1/2 + \zeta$ in this lemma. Hence, for $k \geq 9$, we have that

$$\mathsf{Pr}[Z_0 = Z_1] \geq 1/2 + \frac{\zeta^3}{2} - \frac{\zeta^3 + 1}{k}. \tag{2.2}$$

We must also compute the probability that the simulation halts. This can happen if rewinding fails, i.e., all of the extra challenge-response pairs in the "pool" are used up. For simplicity, we can bound this above by $q2^{-u}$, namely the probability that any single rewinding results in the discarding of $u$ total pairs from the "pool" , where $u = \log_2 q + 1$. Let us set $u = (\log_2 q + 1)$. It follows that we can bound the halting probability above by $q2^{-u} = q2^{-(\log_2 q + 1)} = 1/2$.

Given this bound and eq. 2.2, it is the case that

$$\mathsf{Adv}^{HB^+}_{\mathcal{A}^+}(k, \eta, q, t'_1, t'_2) \geq \frac{1}{2} + \frac{\zeta^3}{4} - \frac{\zeta^3 + 1}{2k},$$

for $t'_1 = kt_1q(2 + \log_2 q)$ and $t'_2 = 2kt_2$. These runtimes are due to the fact that $\mathcal{A}^+_{query}$ may have to do $r$ re-windings for each of $k$ bits. We'll upper bound the cost of each "rewind" with the cost of a complete invocation of $\mathcal{A}_{query}$. Similarly, $\mathcal{A}^+_{clone}$ will run two copies of $\mathcal{A}_{clone}$ for each of $k$ bits. Note that to achieve a positive advantage in the reduction, we need $\zeta^3 > \frac{2}{k-2}$. When the advantage $\zeta$ is small, however, it can be boosted using the standard technique of executing the $\mathcal{A}^+$ multiple times and taking the majority output. (End of Lemma 2 proof.) □

### 2.4.5 Reduction of LPN to HB$^+$-attack

Combining Lemmas 1 and 2 yields a concrete reduction of the LPN problem to the HB$^+$-attack experiment. Given an adversary that has an $\epsilon$-advantage against the HB$^+$-attack experiment within a specific amount of time and queries, there exists an adversary that solves the LPN problem within a concrete upper bound of time and queries. The following theorem follows directly from Lemmas 1 and 2.

**Theorem 1** *Let* $\mathsf{Adv}^{HB^+}(k, \eta, q, t_1, t_2) = \zeta$*, where* $U$ *is a uniform distribution over binary matrices* $\mathbb{Z}_2^{q \times k}$*, and let* $\mathcal{A}$ *be an adversary that achieves this* $\zeta$*-advantage. Then there is an algorithm that can solve a random* $q' \times k$ *instance of the LPN problem*

*in time* $(t'_1, t'_2)$ *with probability* $\frac{1}{k}$*, where* $t'_1 \leq k^2 Lq(2 + \log_2 q)t_1$*,* $t'_2 \leq 2k^2 Lt_2$*,* $q' \leq kLq(2 + \log_2 q)$*,* $\epsilon = \frac{\zeta^3(k-2)-2}{4k}$*, and* $L = \frac{8(\ln k - \ln\ln k)}{(1-2\eta)^2}(\frac{1+\epsilon}{\epsilon})^2$*.*

To put this in asymptotic terms, the LPN problem may be solved by an adversary where $\mathsf{Adv}^{HB^+}(k, \eta, q, t_1, t_2) = \zeta$ in time $O(\frac{(k^5 \log k)(q \log q)\ t}{(1-2\eta)^2\zeta^6})$, where $t = t_1 + t_2$. Note that these concrete bounds are fairly loose. The LPN extractor presented in Section 2.4.3 uses a "fresh" HB sample in each trial. This may be unnecessary, since it could be possible to reuse a single sample many times. A tighter reduction might increase the effective key lengths that will be presented in Section 2.8.

## 2.5 Man-in-the-Middle Attacks

$HB^+$ is vulnerable to a linear-time man-in-the-middle attack first observed by Gilbert, Robshaw, and Sibert [49]. This attack is quite simple: a man-in-the-middle will flip the same bit of either the challenge $\mathbf{a}$ or blinding factor $\mathbf{b}$ during all $q$ rounds of an $HB^+$ authentication. The adversary then observes whether the reader accepts the tag as authentic.

If the reader rejects a tag that *should* be accepted, the adversary knows that the single bit of $\mathbf{a}$ or $\mathbf{b}$ that was changed should have been part of the noisy parity bit response. In other words, the corresponding bit of the secret $\mathbf{x}$ or $\mathbf{y}$ is a 1 bit.

By forcing a legitimate tag to fail $k$ authentication rounds, a man-in-the-middle can completely extract a tag's secrets. This attack does not violate the security proofs presented in this work, because in that setting an adversary does not have access to a reader oracle $\mathcal{R}$ during the query phase of the experiment $HB^+$-attack.

Rather, the security experiments depicted in figure 2-4 specify a "detection model" of anti-counterfeiting. The goal of the adversary in these experiments is to insert a counterfeit tag into the system without detection by the reader. In contrast, in a "prevention" model, an adversary could not create a counterfeit tag under any circumstances.

This weakened model merits some explanation. It may be viewed as defining a *detection-based* authentication system where the adversary is presumed to have a particular aim: to insert a bogus tag into the system without detection. In other words, if an authentication session fails, and $\mathcal{R}$ thus detects an ostensibly counterfeit tag, the adversary is considered unsuccessful. This is equivalent to saying that during the query phase of $HB^+$-attack, the adversary can only initiate or observe successful authentication sessions. Such sessions reveal only information that the adversary can learn directly from the tag, i.e., that access to $\mathcal{R}$ furnishes no additional information.

It is instructive to compare a detection-based model against a *prevention-based* model where the adversary has unfettered access to oracles for $\mathcal{T}$ and $\mathcal{R}$. The aim in a prevention-based model is to ensure against tag cloning irrespective of whether or not an adversary is detected in a counterfeiting attempt against a tag.

A detection-based model is natural and useful in centralized RFID systems, such as those that might be employed with RFID-tagged casino chips or proximity cards, or

in tightly integrated supply chains. In most real-life situations, a man-in-the-middle attack as described by Gilbert, Sibert and Robshaw [49] is impractical, especially in the RFID setting. Besides needing to be physically coupled with a pervasive device, the man-in-the-middle would trigger hundreds of failed authentications to extract a key of a single device.

In such environments, a failed authentication attempt – such as an attempt at counterfeiting – would naturally trigger an alert. RFID tags have an important physical dimension, namely that an HB$^+$ attacker must have some physical presence or proxy to mount an attack. Thus detection has a value in RFID systems not present in general communication networks where an attacker may operate remotely.

To limit leakage of tag secrets, a detection-based authentication system can employ throttling or a lockout in the face of multiple failed authentication attempts. (Of course, any such policy must be constructed carefully to account for the possibility of denial-of-service attacks.) Each authentication attempt can in principle leak up to at most a single bit of information about the secret contained in a tag, as a reader either accepts or rejects at the conclusion of a session.

### 2.5.1 Security Against Man-in-the-Middle: HB$^{++}$

Of course, a stronger prevention-based model is more desirable than a detection-based model. Bringer, Chabanne, and Dottax propose a prevention-based HB$^{++}$, purported to be secure against man-in-the-middle attacks [15]. The main idea behind HB$^{++}$ is to generate a second noisy parity bit $z'$ based on two additional secrets $\mathbf{x}'$ and $\mathbf{y}'$.

The parity bit $z'$ will be generated by bit-rotating permutations of the challenges $\mathbf{a}$ and blinding factors $\mathbf{b}$. That is, in the $i$th round, rotate $i$ bits in each of the permutations $f(\mathbf{a})$ and $f(\mathbf{b})$ to obtain $\mathbf{a}' = rot(f(\mathbf{a}), i)$ and $\mathbf{b}' = rot(f(\mathbf{b}), i)$. The second noisy parity bit $z' = \mathbf{a}' \cdot \mathbf{x}' \oplus \mathbf{b}' \cdot \mathbf{y}' \oplus \nu'$, where $\nu'$ is a second, independent noise bit. HB$^{++}$ is illustrated in figure 2-5.

The idea behind HB$^{++}$ is that the two noisy parity bits will $z$ and $z'$ will act as consistency checks. By flipping a single bit of $\mathbf{a}$ or $\mathbf{b}$, a man-in-the-middle will likely induce an error in either $z$ or $z'$. By observing a reader rejecting an authentic tag, the man-in-the-middle will not know which noisy parity bit caused it, and thus will not learn information about the respective secrets.

Unfortunately, at the time of writing HB$^{++}$ lacks a thorough proof of security and has not yet appeared in publication (although it apparently has been accepted for publication in [15]). This author makes no claim on the security of HB$^{++}$ against man-in-the-middle attacks. It is included here for completeness.

## 2.6 HB$^+$ Optimizations

HB$^+$ carries a large communication round complexity, since a total of $3q$ messages are sent for each authentication. This motivates two optimized versions: *parallel* and *two-round* HB$^+$.

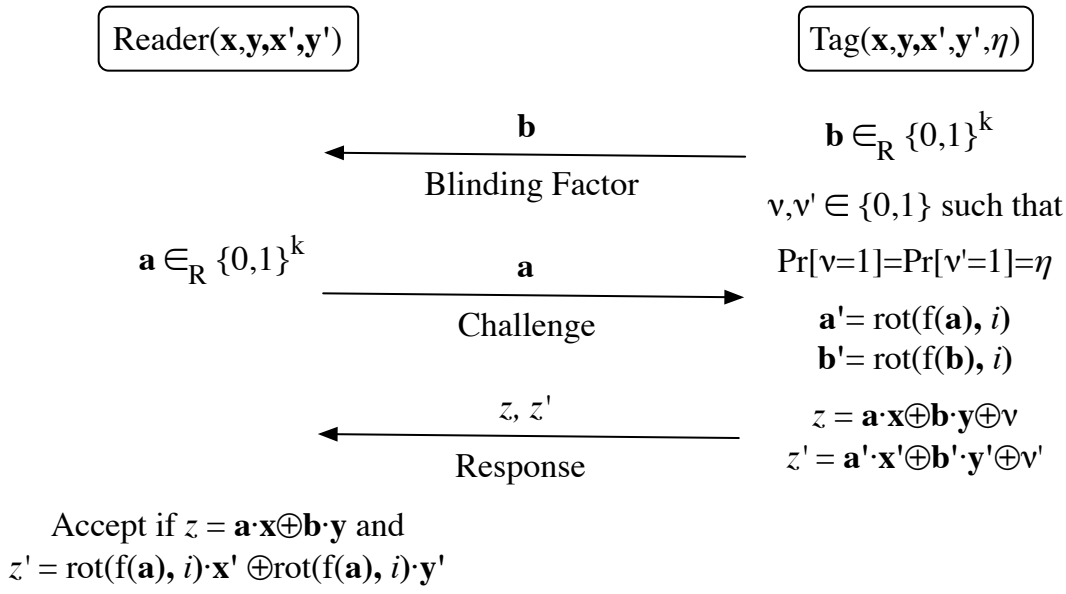


Figure 2-5: Bringer, Chabanne, and Dottax's HB$^{++}$ protocol

The parallel version is quite natural. Why not send all the blinding and challenge vectors together as a single $q \times k$ matrix? All $q$ rounds of the protocol are essentially performed in parallel, so only 3 messages of maximum size $kq$-bits are sent. This protocol is illustrated in figure 2-6.

There was originally a concern with the parallel version of HB$^+$ since an adversarial reader is able to condition its choice of challenges on all the blinding factors. In the sequential-round version, each challenge vector sent by a malicious adversary can only depend on the transcript of its interaction with a tag so far. There might be a chance that an active adversary somehow derives more power from seeing all blinding factors $\mathbf{B}$ up front.

Fortunately, this is not the case. Katz and Shin show that a parallel version of HB$^+$ is, in fact, secure [69]. They also show that HB$^+$ is secure under concurrent composition. Katz and Shin also simplify several steps of the security proofs presented in this chapter by making use of a recent result due to Regev [100], that reduces the LPN problem to solving the Shortest-Vector problem (SVP), albeit requiring the use of a quantum computational step. Regev conjectures that this quantum assumption may be eliminated and that the SVP can be reduced to the LPN problem in the standard model.

A second idea to reduce communication costs is for a tag to send its blinding factor *with* the noisy parity bit response. Figure 2-7 illustrates a sequential, two-round HB$^+$ variant. Combined with Katz and Shin's proof of security under parallel composition, could potentially reduce the number of rounds required for HB$^+$ to 2.

Unfortunately, it is unclear whether this variant is secure. A malicious tag can choose a blinding factor $\mathbf{b}$ that is conditioned on a challenge $\mathbf{a}$. Although it is unknown whether this could potentially lead to a security break, it would allow a malicious tag

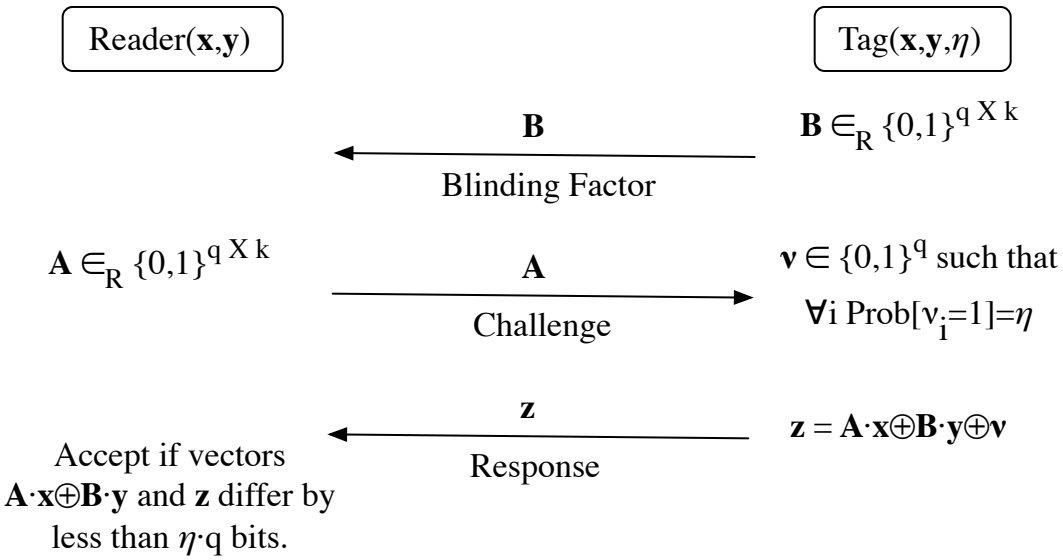


Figure 2-6: A parallel version of $HB^+$

with some partial information on the secrets $\mathbf{x}$ and $\mathbf{y}$ to successfully answer queries.

For instance, suppose a malicious tag somehow obtains the value $\mathbf{x} \oplus \mathbf{y}$. Then given some challenge $\mathbf{a}$, the tag could choose $\mathbf{b} = \mathbf{a}$ and return the value $\mathbf{a} \cdot (\mathbf{x} \oplus \mathbf{y})$, which would be correct. Although a reader could easily detect this attack, a tag might know some different partial data on $\mathbf{x}$ and $\mathbf{y}$ that allows it to respond without detection.

An interesting observation in this example is that this malicious adversary *cannot* be used as a black box to extract the $HB^+$ secrets, although its *a priori* knowledge might be extracted. Another observation is that if an active adversary is able to extract any linear relationship between the two secrets during the challenge phase, they would be able to successfully answer the second phase. The difficultly of that problem is unknown. It may be necessary to rely on non-black box proof techniques in order to prove a two-round $HB^+$ variant is secure.

## 2.7 Anti-Collision and Identification with $HB^+$

A reader initiating an $HB^+$ protocol may not have *a priori* knowledge about which tags it is communicating with. Readers may also operate in environments containing many tags owned by different parties. Simultaneous response to a reader query may cause collisions on the wireless communication channel. Fortunately, $HB^+$ authentication lends itself to both *anti-collision* and *identification*.

Individual tags may be *singulated* from a population using a randomized tree-walking protocol, as described by the author in [120] and [122]. Implementing a randomized tree-walking protocol on a tag requires a random number generator, which conveniently would already be in place for generating blinding factors and noise.

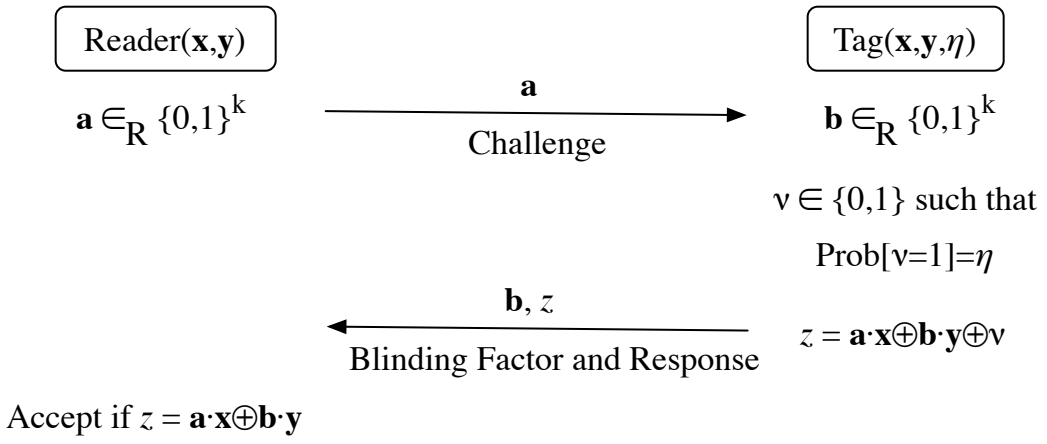


Figure 2-7: A two-round version of HB$^+$

Alternatively, a slotted Aloha anti-collision protocol, as used in wireless 802.11 (WiFi) networks, may be deployed [9, 84]. A slotted Aloha-style protocol is specified by the EPCGlobal generation 2 tag standards [39] and would also make use the random number generator already implemented for the HB$^+$ protocol.

Even when a tag is singulated, a reader will not necessarily know *which* tag it is communicating with. In other words, it may not know which secret to check HB$^+$ response noisy parity bits against. Fortunately, it can "whittle" candidates from its known-tag database after each round of HB$^+$ until it either discovers which tag it is communicating with, or determines that it is communicating with an unknown tag.

To do so, a reader can compare an unidentified tag's noisy parity bit response $z$ against *all* its known tag secrets. (This would be infeasible for an enterprise with billions of RFID devices, but could be done by individuals or smaller enterprises.) Database entries with different secrets will match an expected half of the rounds. After particular entry has failed above a certain threshold, it may be removed from consideration or "whittled". If the unidentified tag's secret is in the reader's database, it will match an expected $(1 - \eta)$ fraction of the rounds. At that point, a reader will have both identified which tag it is communicating with and authenticated it as valid.

## 2.8 Lower Bounds on Key Sizes

This section considers the practical implications of the concrete security bounds implied by Theorem 1. Similar to security constructions based on factoring or finding discrete logarithms, the length of keys that are secure in practice will depend on the state of the art of algorithms and hardware. As a baseline for for comparison, in 1993 DIMACS issued a set of random LPN instances reduced to CNF formulas as a satisfiability problem challenge [63]. These challenge problems used a key length of $k = 32$, $q = 64$ queries, and a noise parameter of $\eta = \frac{1}{8}$.

Solutions were found several years later by specialized exhaustive search algorithms [54, 119]. As an measure of practical hardness, each instance of the DIMACS

challenge [63] took approximately 5-10 minutes to solve on a 200 MHz SGI R10k processor using Warners and van Maaren's search algorithm [119]. Although this search algorithm does not necessarily find the same $\mathbf{x}$ used to generate the responses, it is clear that with a key as short as 32 bits, a small set of samples can be trivially broken on a home PC.

The runtimes in Table 2.2 are based on a concrete analysis of the best known LPN learning algorithm due to Blum, Kalai, and Wasserman (BKW) [11]. As mentioned, this algorithm has an asymptotic runtime of $2^{O(\frac{k}{\log k})}$.

These runtimes should be taken with a grain of salt, since optimizations to the best known algorithm or a tighter reduction may yield much lower constant factors. In fact, Levieil and Fouque (LF) recently reduced the constant factors of the BKW algorithm, which would increase the necessary key lengths in practice [74]. These runtimes are included in Table 2.2.

However, it should be noted that both BKW and LF require an exponential number of samples. In reality, these would be sampled from weak RFID devices which would be a major bottleneck to actually trying to carry out these attacks in practice. Thus, the runtimes given in Table 2.2 *assume an adversary obtain an exponential number samples from an RFID device.*

The BKW algorithm essentially performs Gaussian elimination on a large set of noisy samples, except minimizes the number of linear combinations, which minimizes the noise accumulated throughout the algorithm. By repeating randomized trials, the BKW algorithm can produce the secret $\mathbf{x}$ with high probability.

Omitting details of their algorithm, for $\alpha\beta \geq k$, given $q = \alpha^3 m 2^\beta$ queries and running in $t = C\alpha^3 m 2^\beta$ time, where $m = \max\{\left(\frac{1}{1-2\eta}\right)^{2^\alpha}, \beta\}$, the BKW algorithm can correctly extract $\mathbf{x}$ with an error that is negligible in $k$. In other words:

$$\mathsf{Adv}_{BKW}^{extract-\mathbf{x}}(k, \eta, q, t) \approx 1 - negl(k)$$

Suppose $\eta = 1/4$. Then $m = 2^{2^\alpha}$. If let $k = 224$, then the values $\alpha = 4$ and $\beta = 58$ minimize the value of $t$ necessary to completely reveal a 224-bit secret $x$ with high probability. For these values of $\alpha$ and $\beta$, it is the case that $C\alpha^3 2^{2^\alpha+\beta} \geq 2^{80}$. Thus, a 224-bit LPN secret $x$ with noise parameter $\eta = 1/4$ is secure against an adversary running the improved BKW algorithm that can run for $2^{80}$ steps.

Just for comparison, if $k = 32$, as in the DIMACS challenge, the values that minimize $t$ are $\alpha = 2$ and $\beta = 16$. This yields a $t \approx 2^{24}$. This could reasonably be solved in 10 minutes on a modern processor, as were the DIMACS challenges [54, 119].

As mentioned, these runtimes are simply a reflection of the cost of the best known algorithm. With performance improvements or a tighter analysis, it is likely tha($2^{38}$)t the effective key-length of LPN keys are even shorter. In the meantime, these lengths are still a practical range for low-cost devices and can offer adequate security in many settings.

## 2.9 Practical Implementation Costs

| Key Length | BKW Runtime | LF Runtime |
|---|---|---|
| 32 | $2^{24}$ | $(2^{24})$ |
| 64 | $2^{35}$ | $(2^{28})$ |
| 96 | $2^{46}$ | $2^{33}$ |
| 128 | $2^{56}$ | $(2^{38})$ |
| 160 | $2^{64}$ | $(2^{43})$ |
| 192 | $2^{72}$ | $(2^{47})$ |
| 224 | $2^{80}$ | $2^{52}$ |
| 256 | $2^{88}$ | $(2^{56})$ |
| 288 | $2^{96}$ | $(2^{60})$ |

Table 2.2: Time to extract LPN keys of various length using BKW and LF algorithms. Note that these runtimes assume an adversary has access to an exponential number of samples from a weak device. Estimated runtimes are shown in parenthesis.

Although it could conceivably be implemented in any computing platform (or even carried out by a patient human armed with a coin to flip), the HB$^+$ protocol is quite appropriate for low-cost devices. For instance, the hypothetical EPC tag specified in Table 2.1 could conceivably implement HB$^+$ with perhaps a 256-bit key stored in read-only memory. If one considers the secret key to be the tag's identity itself, this represents no additional cost over a standard device.

The actual HB$^+$ protocol requires very few circuit gates. Computing parity requires only a single XOR and a single AND gate. A single bit register is needed to store the parity as it is being computed. Another register is needed to store challenges and blinding factors.

Fortunately, only one bit of both challenges and blinding factors is needed at any time. Therefore, only a single-bit buffer is needed to temporarily store challenge bits as they arrive from a communication channel, or blinding factor bits as they are generated internally.

A very small amount of control logic is needed for HB$^+$. This would essentially consist of a few bits of protocol state and an $\log k$-bit counter to iterate through a stored secret. It is quite likely that *HB$^+$ may be implemented in under 200 gates of logic*. For comparison, implementing DES or SHA-1 normally requires on the order of 20,000-30,000 gates [41, 42] and highly optimized AES optimizations require approximately 5,000 gates [43].

## 2.10 Conclusion and Open Questions

This chapter presented a new authentication protocol named HB$^+$ that is appropriate for low-cost pervasive computing devices. The HB$^+$ protocol is secure in the presence of both passive and active adversaries and should be implementable within the tight resource constraints of today's EPC-type RFID tags. A number of essential open questions remain, however, before the HB$^+$ can see practical realization.

Above all, the security of the $HB^+$ protocol is based on the LPN problem, whose hardness over random instances remains an open question. It is also an open question whether the two-round variant of $HB^+$ described in Section 2.6 is secure.

As explained in Section 2.5, $HB^+$ is vulnerable to a simple man-in-the-middle attack. This attack assumes that a man-in-the-middle is able to tell whether a particular reader accepted or rejected a tag. To account for this attack, this chapter defines a "detection model", where active adversaries do not have access to reader oracles.

This model is quite reasonable in a real-world setting. Failed authentications will be detected by legitimate readers and, besides, conducting live man-in-them-middle attacks between physically coupled hardware is quite difficult. This detection model is consistent with the strong privacy model proposed by Juels and Weis [68].

Regardless, it is an open question whether an LPN-based protocol may be made secure against a man-in-the-middle attack. The security of the $HB^{++}$ protocol described in section 2.5.1 and in [15], purported to be secure against man-in-the-middle attacks, remains to be seen.

The results in this chapter do not offer direct practical guidance for parameters useful in real RFID tags, something essential for real-world implementation. It would be desirable to see a much tighter concrete reduction than given in this chapter. One avenue might be improvement to the Blum et al. reduction. As mentioned in Section 2.4.3, the efficiency of the modified concrete version of Blum et al. reduction [10] may be improved.

The version in this chapter uses sample values only once. It may be possible to use a single sample to generate several trials per column, or perhaps to generate trials for every column. Reusing sample values could lower the concrete query costs. It is unclear, however, whether the reduction holds over non-uniform input distributions.

It is also unknown what good choices for the noise parameter $\eta$ are. Choosing $\eta$ values close to 0 will leak more information on the true parity values, while values close to $1/2$ will require more protocol rounds. It is likely that for a particular key length, there is an optimal choice for $\eta$ against the best-known BKW algorithm. How to find this optimization is an open question.

Finally, there is second human authentication protocol by Hopper and Blum, based on the "Sum of $k$ Mins" problem and error-correcting challenges [12, 62]. Unlike the HB protocol, this protocol is already supposed to be secure against active adversaries. However, the hardness of the "Sum of $k$ Mins" has not been studied as much as the LPN problem, nor is it clear whether this protocol can efficiently be adapted for low-cost devices. These questions remain open avenues of research.

# Chapter 3

# Commutative and Cascadable Cryptography

Performing sequential or *cascaded* encryption and decryption operations is an intuitive and useful ides used in many applications. For instance, triple-DES (3DES) encryption performs three sequential DES encryption operations under different keys [4]. Public-key encryption operations are often cascaded as well. For example, messages are sequentially encrypted under different public keys in both mix networks for electronic voting [25] and privacy-enhancing "onion routing" [52].

Cascadable cryptosystems are a particular type of multiple encryption system. Multiple encryption systems encrypt data under several keys and possibly under different underlying cryptosystems. In general, a multiple cryptosystem does not necessarily specify the order of decryption operations or define intermediate states of partial decryption. The only requirement is that all keys used to encrypt a message must also be used to decrypt it. In cascadable cryptosystems, one may arbitrarily encrypt an existing ciphertext, or decrypt a ciphertext with a valid key. (Which keys are "valid" for a ciphertext will depend on the specific cryptosystem.) This chapter will focus on cascadable cryptosystems involving a single underlying encryption operator, as opposed to hybrid cryptosystems using several independent encryption schemes.

Unfortunately, standard formal security definitions do not fully capture adversarial abilities in these settings. Adversaries performing chosen-plaintext or chosen-ciphertext attacks may be able to obtain chosen plaintexts or chosen ciphertexts under a cascade of operations, rather than a single operation as in traditional settings. Adversaries may be able to distinguish some message-independent *history* about the operations that produced a particular ciphertext. To address the absence of appropriate security definitions, this chapter formally defines cascadable semantic security and introduces a new notion of historical security.

The most basic cascadable cryptosystem has intuitive properties: one must decrypt in the opposite order of encryption operations. A plaintext encrypted under a sequence of keys $x, y$, denoted as $c = \mathsf{e}_y(\mathsf{e}_x(m))$, must be decrypted under the corresponding keys in reverse order, i.e. $m = \mathsf{d}_x(\mathsf{d}_y(c))$. For convenience, parentheses will be omitted. The aforementioned sequence of encryption operations evaluated on a

message $m$ would be represented by $\mathsf{e}_y\mathsf{e}_x(m)$. However, other classes of cascadable cryptosystems may allow different decryption orders.

A particularly useful and interesting class of cascadable cryptosystems are those that exhibit commutative properties. In commutative cryptosystems, one may decrypt with any key that a ciphertext has already been encrypted with. A ciphertext $c = \mathsf{e}_y\mathsf{e}_x(m)$ may be decrypted as either $m = \mathsf{d}_y\mathsf{d}_x(c)$ or $m = \mathsf{d}_x\mathsf{d}_y(c)$.

Commutative cryptosystems, such as Pohlig-Hellman [99] and Massey-Omura [78], have existed for over 25 years and are the basis of many proposed applications. For instance, Shamir, Rivest, and Adleman's classic "Mental Poker" and three-pass key exchange protocols both rely on commutativity [105, 106]. More recently, Agrawal, Evfimievski, and Sriakant [1], and Clifton et al. [27] present data mining and private set intersection applications based on commutativity.

To illustrate an application of commutativity, consider the three-pass key exchange protocol illustrated in figure 3-1. In this protocol, Alice and Bob each have respective secret keys $a$ and $b$. Alice wishes to share a secret $s$ with Bob. If Alice and Bob have a cascadable, commutative cryptosystem at their disposal, they can engage in the following protocol:

THREE-PASS KEY EXCHANGE

1. Given input $s$, Alice generates a random key $a$ and sends Bob $\mathsf{e}_a(s)$.
2. Bob generates a random key $b$ and sends Alice $\mathsf{e}_b\mathsf{e}_a(s) \simeq \mathsf{e}_a\mathsf{e}_b(s)$.
3. Alice sends Bob $\mathsf{d}_a\mathsf{e}_a\mathsf{e}_b(s) = \mathsf{e}_b(s)$.
4. Bob computes $\mathsf{d}_b\mathsf{e}_b(s) = s$.

Figure 3-1: Three-pass key exchange based on commutativity

Commutativity is just one of many possible properties exhibited by cascadable cryptosystems. To model these various flavors, the new cascadable cryptosystem definition in this work incorporates *string rewrite systems*. String rewrite systems model cryptographic operations as strings of symbols and capture the interaction between various symbols with rewrite rules. Rewrite systems will be a useful and flexible tool that may model a wide variety of cryptosystems.

**Organization.** Section 3.1 offers a brief primer on rewrite systems. Similar rewrite models may be applied toward developing security definitions for re-randomizable, homomorphic, or threshold cryptographic systems. Section 3.2 will formally define cascadable cryptosystems and a notion of efficiently cascadable cryptosystems. Section 3.3 formally defines cascadable semantic security and introduces a new notion of historical security. Section 3.4 presents two cascadable semantically secure cryptosystems. One construction is built from a generic semantically secure cryptosystem. The

second is a new, cascadable semantically secure, commutative public-key cryptosystem. This construction is based on an arbitrary, semantically secure cryptosystem that supports homomorphic multiplication, such as ElGamal.

## 3.1 String Rewrite System Primer

In order to discuss cascadable cryptosystems, it is necessary to have the language and tools to describe sequences of cryptographic operations. A simple way of representing these operations is with strings of symbols. As alluded to in figure 3-1, encrypting or decrypting under a key $k$ may be represented by $\mathsf{e}_k$ and $\mathsf{d}_k$ symbols.

Different sequences of operations represented by strings may yield equivalent distributions when actually evaluated on a message. To model this notion of string equivalence, this section will develop a set of *rewrite rules* that allow one to transform one string of operations to another.

Thus, string rewrite systems (SRS) will be used to model cascaded cryptographic operations. Similar rewrite systems were used by Dolev and Yao [38], and Dolev, Even, and Karp [37] to model sequential cryptographic operations in two-party protocols. Different SRS models may capture different properties, for instance commutativity.

Several cryptosystems are based on the hardness of various rewrite problems. The works of Wagner and Magyari [117], Oleshchuk [93], and Samuel et al. [103] are all examples of rewrite systems being used as hard underlying problems. This chapter does not apply rewrite systems in this manner, and instead uses them for analysis along the lines of Dolev et al.

This section provides a brief primer on rewrite system concepts that are relevant to this chapter. This is by no means a thorough review of the topic, as rewrite systems are the subject of a rich and broad body of literature. Surveys by Dershowitz, Jouannaud [33], and Plaisted [34] offer a much more extensive introduction to rewrite systems.

**Symbols, Alphabets, and Strings.** As in the three-pass key exchange protocol from figure 3-1, encryption and decryption operations under a key $k$ are respectively represented by symbols $\mathsf{e}_k$ and $\mathsf{d}_k$. The set of symbols for encryption or decryption under every possible key is defined to be the *alphabet* $\Gamma$.

Strings are elements in $\Gamma^*$. Symbols in $\Gamma$ may represent operations in either public-key or symmetric cryptosystems that may be either deterministic or randomized. This chapter exclusively focuses on randomized, public-key operations. Both encryption with public key $pk$ and decryption with public key $sk$ will use $pk$ as a symbolic index, i.e. $\mathsf{e}_{pk}$ and $\mathsf{d}_{pk}$. This allows someone to reference a decryption operation, without necessarily knowing the secret key.

**Rewrite Rules.** A rewrite rule denoted as $x \to y$ specifies that a string $x$ may be replaced, wherever it appears, by a second string $y$, but not necessarily vice versa.

Given a string $s = \alpha x \beta$, applying the rule $x \to y$ yields the string $t = \alpha y \beta$. For example, if $\Gamma = \{\mathsf{e}, \mathsf{d}\}$, applying the rule $\mathsf{de} \to \epsilon$ to the string $s = \mathsf{edde}$ yields $t = \mathsf{ed}$.

**String Rewrite Systems.** A string rewrite system (SRS) is defined as a pair $(\Gamma, R)$, consisting of an alphabet $\Gamma$ and a set of rules $R$. As an example, consider the SRS modeling cascadable cryptographic operations, similar to the rewrite systems used by Dolev and Yao [38] and Dolev, Even, and Karp [37]:

**Definition 2 (Cascadable Cryptosystem Model)** *Let $K$ be the set of all possible keys and let $\epsilon$ represent the empty string. Define the string rewrite system $\mathcal{S}_{casc} = (\Gamma, R)$ as follows:*

$$\Gamma = \bigcup_{k \in K} \{\mathsf{e}_k, \mathsf{d}_k\}, \quad R = \bigcup_{k \in K} \{\mathsf{d}_k \mathsf{e}_k \to \epsilon\}$$

The alphabet in $\mathcal{S}_{casc}$ contains a symbol for encryption and decryption under every possible key. Strings in $\mathcal{S}_{casc}$ represent sequences of operations that may be evaluated on a plaintext. For instance, the string $\mathsf{e}_k \mathsf{e}_j$ models the operation $m \mapsto E(k, E(j, m))$, for an arbitrary $m$. Rules of the form $\mathsf{d}_k \mathsf{e}_k \to \epsilon$ in $\mathcal{S}_{casc}$ model decryption operations.

**Derivability.** If there exists some sequence of rules that when applied to a string $s$, yield a string $t$, then $s$ derives $t$, or alternatively, $t$ is derived from $s$. This derivability relation is denoted as $s \xrightarrow{*} t$. For example in $\mathcal{S}_{casc}$, the string $\mathsf{d}_j \mathsf{d}_k \mathsf{e}_k \mathsf{e}_j \xrightarrow{*} \epsilon$, since $\mathsf{d}_j \mathsf{d}_k \mathsf{e}_k \mathsf{e}_j \to \mathsf{d}_j \mathsf{e}_j$ and $\mathsf{d}_j \mathsf{e}_j \to \epsilon$.

**Normal Strings.** A string in a SRS $(\Gamma, R)$ is *normal* (or *irreducible*) if there do not exist any rules in $R$ which can be applied to it. In $\mathcal{S}_{casc}$, the strings $\mathsf{e}_k$, $\mathsf{e}_j$, and $\mathsf{d}_k$ are each normal, while $\mathsf{d}_k \mathsf{e}_k$ is not. Normal strings are relevant to the definition of string equivalence.

**Derivation Cycles.** A SRS may have two strings $s$ and $t$ such that $s$ derives $t$ and $t$ derives $s$, i.e. $s \xrightarrow{*} t$ and $t \xrightarrow{*} s$. This is referred to as a *derivation cycle* and is denoted $s \overset{*}{\leftrightarrow} t$.

**Termination.** A SRS is *terminating* (or *Noetherian*) if no derivation cycles exist. In other words, only a finite number of rules may be applied to any string in $\Gamma^*$ before a normal string is derived. Normal forms are not necessarily unique. A string $s$ may have different normal forms that are derivable by applying different sequences of rules. Termination is relevant to the definition of string equivalence.

**Confluence.** *Confluent* string rewrite systems have the property that any string's derivations must share a common derivation. That is, in a confluent system, given $s$, $s \xrightarrow{*} t$, and $s \xrightarrow{*} u$, there must exist some $w$ such that $t \xrightarrow{*} w$ and $u \xrightarrow{*} w$. In string rewrite systems, confluence is equivalent to what is called the *Church-Rosser* property, and is relevant to the definition of equivalence.

**Canonical Forms.** *Canonical* string rewrite systems are both terminating and confluent. In a canonical SRS, every string has a unique normal form, called its *canonical form*. Given a string $s$ in a canonical system, its canonical form is denoted as $\bar{s}$. By inspection, the cascadable SRS $\mathcal{S}_{casc}$ from definition 2 is a canonical SRS.

**String Equivalence.** This work defines two strings in a canonical SRS to be *equivalent* if and only if they have the same canonical form, i.e $(s \simeq t) \iff (\bar{s} = \bar{t})$ [1]. In $\mathcal{S}_{casc}$ the strings $s = \mathsf{e}_k$ and $t = \mathsf{d}_j\mathsf{e}_j\mathsf{e}_k$ are equivalent because $\bar{s} = \mathsf{e}_k = \bar{t}$. However, the string $u = \mathsf{d}_k\mathsf{e}_k\mathsf{e}_j$ is not equivalent to either, because its canonical form is $\bar{u} = \mathsf{e}_j$.

**Strings versus Terms.** Rewrite system literature often deals with more general *term rewrite systems*. Rather than simply symbols from $\Gamma$, terms may also contain variables representing arbitrary terms. Term rewrite systems can model a richer set of cryptographic properties like re-randomization, threshold operations, homomorphic operations, signatures, hashing, etc. For instance, one might model an additive homomorphic property with a term rewrite rule $\mathsf{e}_k(x) * \mathsf{e}_k(y) \rightarrow \mathsf{e}_k(x + y)$.

The definitions and concepts in this chapter are based on string rewrite systems, but might be adapted to a term rewrite model and applied to a broader range of cryptosystems. Adapting the definitions appearing in the following sections to support term rewrite systems is left as an open problem.

## 3.2 Cascadable Cryptosystems

This section first introduces convenient notation, then formally defines cascadable cryptosystems and efficiently cascadable cryptosystems.

### 3.2.1 Notation and Definitions

**Negligible Functions.** The notation $\text{negl}(\cdot)$ to denotes some function such that for all positive polynomials $\text{poly}(\cdot)$ and sufficiently large $\kappa$, it is the case that $\text{negl}(\kappa) < 1/\text{poly}(\kappa)$.

**Evaluating Strings.** A string $s \in \Gamma^*$ represents a sequence of encryption and decryption operations. Each operation in a string may be evaluated in sequential order on some plaintext from a message space $M$. Given a string $s \in \Gamma^*$ and a message $m \in M$, $s(m)$ will represent the evaluation of the operations represented by $s$ from right-to-left on $m$.

Symbols in $s$ may represent randomized operations, so $s(m)$ is defined as a distribution of ciphertexts. The notation $s(m) = t(m)$ means that evaluating strings $s$ and $t$ on message $m$ yields identical distributions. Note that $s(m) = \perp$, where $\perp$ is a

[1] This may be a non-standard notion of equivalence compared to the traditional string rewrite system literature. Equivalence may be defined differently in non-canonical rewrite systems, for example two strings may be defined as equivalent if they belong to the same derivation cycle.

special error symbol is allowable, and will be typical in some systems. For instance, this may be the defined behavior when $m$ is not a valid ciphertext.

**Net Encryption Height.** Given any string $s \in \Gamma^*$, the *net encryption height* function $\nu(s)$ is defined as the minimum length of any string $t$ such that $ts \simeq \epsilon$. If no such $t$ exists, $\nu(s) = \infty$. Define the height of the empty string $\nu(\epsilon) = 0$. Informally, $\nu(s)$ is the minimum number of decryptions needed to recover a ciphertext obtained by evaluating $s$ on some message.

The function $\nu$ is defined with respect to a particular SRS model. In $\mathcal{S}_{casc}$, it is the case that $\nu(\mathsf{e}_k) = 1$ since $\mathsf{d}_k\mathsf{e}_k = \epsilon$. Similarly, $\nu(\mathsf{e}_k\mathsf{e}_j) = 2$. It is also the case that $\nu(\mathsf{d}_k) = \infty$ and $\nu(\mathsf{e}_k\mathsf{d}_k) = \infty$, since there are no strings which may decrypt for $\mathsf{d}_k$ or $\mathsf{e}_k\mathsf{d}_k$. Changing the model will change $\nu$. For instance, if the rules $\mathsf{e}_k\mathsf{d}_k \to \epsilon$ were added to the cascadable SRS, then $\nu(\mathsf{e}_k\mathsf{d}_k) = 0$.

**Cipherspaces.** Let $M$ be the set of all plaintext messages; typically $M = \Sigma^{\ell(\kappa)}$, where $\Sigma$ is some fixed set of symbols (perhaps binary) and $\ell : \mathbb{N} \to \mathbb{N}$ is some fixed function. Let $S_i = \{s \in \Gamma^* | \nu(s) = i\}$. Define $C_i$ to be the set of all possible ciphertexts that can result by evaluating a string with encryption height $i$, i.e. $C_i = \{c | c \in s(m), m \in M, s \in S_i\}$. Define $C_0 = M$.

For some cryptosystems, it may be the case that there is a single cipherspace. It may also be the case that the $C_i$'s are disjoint from each other. Cipherspaces will also be defined with respect to specific strings. Define $C_s = \{c | c \in s(m), m \in M\}$. Define $C_\epsilon = C_0 = M$ and $C_\infty = \bigcup_{s \in \Gamma^*} C_s$.

**Abuse of Set Notation.** This chapter may use the notation $\mathsf{e} \in s$ or $\mathsf{d} \in s$ to indicate that a symbol appears in a string $s$. This is a bit of an abuse of notation since $s$ is a string, and not a set. However, meaning should be clear from context.

### 3.2.2 Cascadable Cryptosystems

This section formally defines cascadable cryptosystems. This definition differs from traditional cryptosystems by explicitly defining how encryption and decryption operations behave over cipherspaces. This requires introducing a new property called *rewrite fidelity*, akin to the completeness property in traditional settings. Rewrite fidelity defines semantic meaning to evaluating symbolic strings on messages.

**Definition 3 (Cascadable Cryptosystem)** *A cascadable cryptosystem $\mathcal{C}$ is a quadruple of probabilistic polynomial time algorithms $(\mathsf{SysGen}, \mathsf{G}, \mathsf{E}, \mathsf{D}, \mathcal{S})$ and a string rewrite system $\mathcal{S} = (\Gamma, R)$ that satisfy the following properties, given a security parameter $\kappa$:*

1. *Message Space: $M = C_0 = \Sigma^{\ell(\kappa)}$ where $\ell : \mathbb{N} \to \mathbb{N}$ is a fixed function.*
2. *System Parameters: $\mathsf{SysGen}(1^\kappa) \to \rho$*
3. *Key Generation: $\mathsf{G}(\rho) \to (pk, sk)$*

4. *Encryption:* $\forall (pk, sk) \in \mathsf{G}(\rho),\ \forall i \geq 0,\ \forall c \in C_i,\ \mathsf{E}(pk, c) \in C_{i+1}$

5. *Decryption:* $\forall (pk, sk) \in \mathsf{G}(\rho),\ \forall i > 0,\ \forall c \in C_i,\ \mathsf{D}(sk, c) \in C_{i-1} \cup \{\perp\}$

6. *Rewrite Fidelity:* $\forall m \in M,\ \forall s \in \Gamma^*,\ s(m) = \bar{s}(m)$.

The function $\mathsf{SysGen}$ outputs some system parameters that are used to generate all keys within a particular instance of the cryptosystem. The key generation function $\mathsf{G}$ is defined as it would be in a traditional cryptosystem.

One detail that needs to be addressed is binding the generated keys $(pk, sk) \in \mathsf{G}(\rho)$ with their respective $\mathsf{e}$ and $\mathsf{d}$ symbols. Define a key pair's "index" to be the value of its public key. Thus, the symbols corresponding to $(pk, sk)$ will be $\mathsf{e}_{pk}$ and $\mathsf{d}_{pk}$. Any party knowing a public key $pk$ can specify its corresponding encryption and decryption symbols, but can only *evaluate* $\mathsf{e}$ symbols on a given input message. Evaluating a $\mathsf{d}$ symbol requires the corresponding $sk$.

Encryption and decryption functions differ somewhat from traditional definitions. In the traditional model, an encryption function may be defined to operate on input from $\Sigma^*$. There is no distinction between arbitrary strings and elements from some cipherspace.

This definition specifies that on a valid input from some cipherspace $C_i$ and a public key $pk$, $\mathsf{E}$ will output a valid element of the next cipherspace $C_{i+1}$. Decryption is more subtle. If given some $c \in s(m)$, $\mathsf{d}_k(c)$ may output $\perp$ if $\nu(d_k s) = \infty$. In $\mathcal{S}_{casc}$, this would be the case if one tried to decrypt with a key that a message was not just encrypted with.

Rewrite fidelity is a critical property of cascadable rewrite systems and is important to both correctness and proofs of security. An important consequence of rewrite fidelity is that $(s \simeq \epsilon) \implies (s(m) = \{m\})$. As one would expect, a string $s$ that is equivalent to doing nothing should return the original plaintext. Another consequence is that $(s \simeq t) \implies (s(m) = t(m))$, which captures the property that two equivalent strings should yield the same distribution of ciphertexts when evaluated on any message.

A cascaded operation may expand the length of successive ciphertexts. Some cascadable cryptosystems may, for example, double the length of the output after each encryption. In that case, cascaded ciphertexts would exponentially grow as a function of $\nu(s)$. We define efficiently cascadable cryptosystems so that the difference in length between an encryption input and output may only be some polynomial of $\kappa$. The difference may not be a function of the input length:

**Definition 4** *[Efficiently Cascadable Cryptosystems] A cascading cryptosystem $\mathcal{C}$ is efficiently cascadable if*

$$\exists c,\ \forall \rho \in \mathsf{SysGen}(1^\kappa),\ \forall (pk, sk) \in \mathsf{G}(\rho),\ \forall x \in \Sigma^*\ |\mathsf{E}(pk, x)| - |x| = O(\kappa^c)$$

## 3.3 Security Definitions

Traditional notions of semantic security do not accommodate commutative cryptosystems, since an adversary may be able to distinguish a ciphertext by some message-independent information, such as which keys successfully decrypt it. (This will be explained in more detail later.) At its core, this incompatibility occurs because cascaded encryption or decryption operations may embed some observable *history* in a ciphertext. This history may contain the number of times a message has been encrypted, which public keys it was encrypted with, which secret keys will successfully decrypt it, or any number of other efficiently computable historical properties.

Because traditional semantic security is defined only over a plaintext message space $M$ and involves only a single encryption, history is not an issue. However, in an indistinguishability under chosen plaintext attack (IND-CPA) experiment with a cascadeable system, an adversary may choose two ciphertexts with different histories as challenge messages, such that it is trivial to distinguish their encryptions.

Essentially, rather than being limited to choosing plaintext messages from $M$ in the IND-CPA experiment, an adversary in a cascadable setting may also choose arbitrary ciphertexts as challenge messages. That is, challenge messages may be from one or more $C_s$ cipherspace domains. To consider semantic security over a specific message domain, we define *semantic security over message domain $D$* as simply semantic security where challenge messages in the IND-CPA experiment are drawn from domain $D$.

This section will define a notion of *cascadable semantic security* that ensures that challenge messages have the same history. It will compare and contrast cascadable semantic security to traditional semantic security over various domains. Section 3.3.3 then discusses a notion of historical security and historical revelation properties. Finally, Section 3.3.4 considers the problem of defining a stronger notion of CCA security in a cascadable model.

### 3.3.1 Cascadable IND-CPA Experiment

This section defines a cascadable indistinguishability under chosen plaintext attack (CIND-CPA) experiment. Denote a cascadable cryptosystem as $\mathcal{C} = (\mathsf{SysGen}, \mathsf{G}, \mathsf{E}, \mathsf{D}, \mathcal{S})$. Denote a probabilistic polynomial time interactive Turning machine adversary as $\mathsf{Adv}$.

In definition, 5, Steps 1 and 2 generate system parameters $\rho$ and a set of $n$ key pairs. This differs from a traditional public-key IND-CPA experiment, as shown in definition 6 where a single key pair would be generated. Step 3 denotes a set of symbols $K$ containing an encryption symbol $\mathsf{e}_{pk}$ for each public key generated in step 2.

In step 4, the adversary is provided all values generated during setup. This adversary may internally compute any of the cryptosystem functions, $\mathsf{SysGen}$, $\mathsf{G}$, $\mathsf{E}$, or $\mathsf{D}$ using any of the public keys it was provided in step 2, or any keys it generates itself. At this point, a traditional IND-CPA experiment would simply output two messages $m_0$ and $m_1$ from the plaintext message space $M$. The cascading CIND-CPA experiment outputs an additional string $s \in \Gamma^*$.

**Definition 5 (CIND-CPA Experiment)** *A cascadable indistinguishability under chosen plaintext attack experiment with adversary* $\mathsf{Adv}$ *and cascadable cryptosystem* $\mathcal{C}$ *is defined as follows:*

$\text{Exp}^{CIND-CPA}_{\mathsf{Adv},\mathcal{C}}[\kappa, n]$*:*

1. $\mathsf{SysGen}(1^\kappa) \to \rho$
2. *Call* $\mathsf{G}(\rho)$ *n times to generate* $(pk_1, sk_1), \ldots, (pk_n, sk_n)$
3. *Denote the key symbol set* $K = \{\mathsf{e}_{pk_1}, \ldots, \mathsf{e}_{pk_n}\}$.
4. $\mathsf{Adv}(\rho, pk_1, \ldots, pk_n)$ *finds* $(m_0, m_1, s)$ *such that:*
   (a) $m_0, m_1 \in M$
   (b) $s \in \Gamma^*$ *such that* $(\infty > \nu(s) > 0) \wedge (\exists \mathsf{e}_{pk})(\mathsf{e}_{pk} \in K \wedge \mathsf{e}_{pk} \in \bar{s})$
5. *Let* $b \xleftarrow{R} \{0, 1\}$ *and* $c \in_R s(m_b)$
6. $\mathsf{Adv}(c)$ *outputs guess bit* $b'$

There are two restrictions on which $s$ values may be output by $\mathsf{Adv}$. First, $\nu(s)$ must be finite and non-zero, meaning there must exist some non-empty string $t$ such that $ts \simeq \epsilon$. This means that trivial values like $s = \epsilon$ or $s = \mathsf{d}_k\mathsf{e}_k$ are forbidden. In $\mathcal{S}_{casc}$, strings like $\mathsf{d}_k$ would be forbidden, since $\nu(\mathsf{d}_k) = \infty$.

The canonical form $\bar{s}$ must also contain at least one encryption symbol corresponding to a key generated in step 2. In other words, $s$ must have at least one net encryption under a public key that $\mathsf{Adv}$ did not generate on its own, i.e., $(\exists \mathsf{e}_{pk})(\mathsf{e}_{pk} \in K \wedge \mathsf{e}_{pk} \in \bar{s})$. Otherwise, $s$ might contain encryptions strictly under keys that $\mathsf{Adv}$ generated itself.

How does this experiment relate to the traditional IND-CPA experiment? Recall the setup of the traditional IND-CPA experiment:

**Definition 6 (IND-CPA Experiment)** *An indistinguishability under chosen plaintext attack experiment with adversary* $\mathsf{Adv}$ *and cryptosystem* $\mathcal{C}$ *is defined as follows:*

$\text{Exp}^{IND-CPA}_{\mathsf{Adv},\mathcal{C}}[\kappa]$*:*

1. $\rho \leftarrow \mathsf{SysGen}(1^\kappa)$
2. $(pk, sk) \leftarrow \mathsf{G}(\rho)$
3. $\mathsf{Adv}(\rho, pk)$ *finds two messages* $(m_0, m_1)$
4. $b \xleftarrow{R} \{0, 1\}$
5. $\mathsf{Adv}(\mathsf{e}_{pk}(m_b))$ *outputs guess bit* $b'$

In the traditional IND-CPA experiment, an adversary given a single public key $pk$ will output two messages $m_0$ and $m_1$ and expects $E(pk, m_b)$ in response. An adversary with a significant advantage in the IND-CPA experiment would be able to trivially break the CIND-CPA experiment by simply selecting $s = \mathsf{e}_{pk}$ for some random $pk \in K$. It would then obtain the exact input it would expect in the IND-CPA experiment, namely $\mathsf{e}_{pk}(m_b)$.

### 3.3.2 Cascadable Semantic Security

This cascadable CIND-CPA experiment $\text{Exp}^{CIND-CPA}_{\mathsf{Adv},\mathcal{C}}$ "succeeds" if $\mathsf{Adv}$ distinguishes the ciphertext $c \in s(m_b)$ and outputs the correct bit $b' = b$.

**Definition 7** *[Cascadable Semantic Security] A cascadable cryptosystem* $\mathcal{C} = (\mathsf{SysGen}, \mathsf{G}, \mathsf{E}, \mathsf{D}, \mathcal{S})$ *is cascadable semantically secure if:*

$$\forall n = \text{poly}(\kappa),\ \forall \mathsf{Adv} \in Time(\text{poly}(\kappa))\ \Pr\left[\text{Exp}^{CIND-CPA}_{\mathsf{Adv},\mathcal{C}}[\kappa, n]\ \textit{succeeds}\ \right] \leq \frac{1}{2} + \text{negl}(\kappa)$$

This cascadable semantic security definition implies several criteria about which string rewrite models could potentially be part of a cascadable semantically secure cryptosystem. For example, the CIND-CPA experiment implies several criteria that a SRS $\mathcal{S}$ must have, such as:

- $\mathcal{S}$ must be canonical, since canonical forms are necessary in the definition of cascadable cryptosystems.
- The string $s$ generated by $\mathsf{Adv}$ in the security experiment must contain at least one $\mathsf{e}$ symbol corresponding to a randomly generated key, thus $\Gamma$ must contain an $\mathsf{e}$ symbol for every possible key generated by $\mathsf{G}$.
- The string $s$ chosen by an adversary must have a finite, non-zero encryption height $\nu(s)$. Thus, $\mathcal{S}$ must contain rules to at least decrypt arbitrary $\mathsf{e}$ symbols.

This by no means characterizes all string rewrite systems which could potentially be part of a cascadable semantically secure cryptosystem. Completely characterizing this class of string rewrite systems is left as an open problem.

**Theorem 2** $\mathcal{C}$ *is cascadable semantically secure* $\implies$ $(\forall s \in \Gamma^*)$ $\mathcal{C}$ *is semantically secure over domain* $C_s$*.*

**Proof:** Recall the definition of $C_s$ is the space of all messages in $M$ encrypted under string $s$. Suppose $\mathcal{C}$ is not semantically secure over messages drawn from some message space $C_s$. Then there exists an adversary able to distinguish encryptions $c \in \mathsf{e}_{pk}(s(m_b))$. Such an adversary could simply output $s' = \mathsf{e}_{pk}s$, $m_0$, and $m_1$ in a CIND-CPA experiment. □

**Theorem 3** *If* $(\forall s \in \Gamma^*)$ $\mathcal{C}$ *with rewrite system* $\mathcal{S}_{casc}$ *is semantically secure over domain* $C_s$ *then* $\mathcal{C}$ *with rewrite system* $\mathcal{S}_{casc}$ *is cascadable semantically secure.*

**Proof:** Suppose that a cryptosystem with rewrite system $\mathcal{S}_{casc}$ is not cascadable semantically secure. Then some adversary outputting $s, m_0, m_1$ will be able to distinguish $c \in s(m_b)$ with some non-negligible advantage. Recall $s$ must contain some $\mathsf{e}_{pk}$ that the adversary did not generate itself, and must have finite $\nu(s) > 0$.

As will be shown in the proof of theorem 6, canonical strings $\bar{s}$ with finite non-zero $\nu(s)$ must consist entirely of $\mathsf{e}$ symbols. Let $\mathsf{e}_{pk}$ be the last (left-most) encryption in $s$ that the adversary did not generate and denote $s = u\mathsf{e}_{pk}v$. Since all symbols in $u$ may be evaluated by an adversary, there exists some polynomial-time adversary

able to distinguish ciphertexts in $\mathsf{e}_{pk}(v(m_b))$. This means that the cryptosystem is *not* semantically secure over $C_v$. □

An open question remains whether this holds for arbitrary rewrite systems. Theorem 3 does not immediately follow if canonical strings might contain $\mathsf{d}$ symbols. A first step would be to characterize which rewrite systems could conceivably be part of a cascadable rewrite system.

The author conjectures that this implication does not hold for arbitrary rewrite systems. There may be artificial cascadable cryptosystem or rewrite models constructions designed specifically not to be semantically secure for some $C_s$.

**Theorem 4** *If $\mathcal{C}$ is cascadable semantically secure it does* not *imply* $(\forall s, t \in \Gamma^* | s \not\simeq t)$ *$\mathcal{C}$ is semantically secure over domain $C_s \cup C_t$.*

**Proof:** The commutative cascadable semantically secure cryptosystem $\mathcal{C}_{com}$ defined in Section 3.5 is not semantically secure over domain $C_\infty$. In the traditional semantic security model, an adversary may generate its own keys $x$ and $y$ and choose message $m_0 \in \mathsf{e}_x(m)$ and $m_1 \in \mathsf{e}_y(m)$. The adversary then obtains $c \in \mathsf{e}_{pk}(m_b)$, and tries to compute $\mathsf{d}_a(c)$ and $\mathsf{d}_b(c)$.

In $\mathcal{C}_{com}$, one of these evaluations will successfully decrypt to $\mathsf{e}_{pk}(m)$, while the other will output $\perp$. Thus, an adversary is distinguishing $c$ by the history of operations that produced it. Since $m_0 \in C_{\mathsf{e}_x}$ and $m_1 \in C_{\mathsf{e}_y}$, $\mathcal{C}_{com}$ is not semantically secure over domain $C_{\mathsf{e}_x} \cup C_{\mathsf{e}_y}$. This obviously implies that $\mathcal{C}_{com}$ is not semantically secure over $\mathcal{C}_\infty$ either. □

### 3.3.3 Historical Security

Cascadable semantic security is a direct analogy of traditional semantic security, but does not capture an important facet of cascadable cryptosystems. Namely, what happens when an adversary is able to distinguish some message-independent *history* of a ciphertext? The essential question of historical security is *what information can an adversary learn about $s$ given $c \in s(m)$?*

The deterministic history of a ciphertext $c \in s(m)$ is entirely defined by the operation string $s$. It is assumed that random inputs used during evaluation are not contained in $s$, which is why this section is concerned only with the deterministic history. Alternative rewrite systems might model randomization operations with symbols like $\mathsf{r}_x$, which could represent re-randomization with a value $x$. However, this work will exclusively focus on deterministic histories and will not consider random inputs used in evaluating $\mathsf{e}$ and $\mathsf{d}$ symbols.

In the cascadable semantic security model, historical security is not an issue since the CIND-CPA experiment adversary chooses $s$ itself. Whatever information that the ciphertext $c \in s(m_b)$ reveals about $s$ is irrelevant. *A cascadable semantically secure cryptosystem might even include $s$ verbatim in its output.* As Theorem 4 illustrates, history *does* matter in the traditional semantic security setting. If an IND-CPA adversary chooses $m_0 \in s(x)$ and $m_1 \in t(x)$, and is able to distinguish

between ciphertexts encrypted under $\mathsf{e}_{pk}s$ or $\mathsf{e}_{pk}t$, then the cryptosystem will not be semantically secure over domain $C_s \cup C_t$. In the commutative example from the proof of Theorem 4, it is the case that $s = \mathsf{e}_x$ and $t = \mathsf{e}_y$.

Why might the ciphertext history matter? There are many settings in which it could benefit an attacker to identify which parties have encrypted a ciphertext, for example mix-nets for electronic voting [25] or onion routing for private communication [52]. Different settings may require a ciphertext to hide different aspects of its history. For example, cascaded encryptions and re-randomization in mix-nets may need to hide the corresponding public keys, which motivated the development of universally re-randomizable encryption [53].

**Historical Revelation Properties.** Which ciphertext historical properties might be revealed to an adversary in a cascadable cryptosystem? As an example, the $\mathcal{C}_{com}$ construction in Section 3.4 reveals the net encryption height $\nu(s)$. For this reason, this construction may be considered "height revealing". An IND-CPA adversary allowed to select challenge messages from $C_s \cup C_t$ where $\nu(s) \neq \nu(t)$ would be able to trivially distinguish encryptions by their length. Traditional semantic security definitions usually avoid this issue by assuming that challenge messages are of the same length.

As another example, a ciphertext $c \in s(m)$ may reveal the set of keys used to encrypt it, i.e. it may reveal $\{\mathsf{e}_k | \mathsf{e}_k \in \bar{s}\}$. This revelation property is informally called "public key set revealing". One may also consider a variant where $K$ is a set of $\mathsf{e}$ symbols, and $c \in s(m)$ reveals $\{\mathsf{e}_{pk} | \mathsf{e}_{pk} \in \bar{s} \wedge \mathsf{e}_{pk} \in K\}$. This property will be called "known public key set revealing".

By contrast, a "known secret key set revealing" cascadable cryptosystem reveals if any public keys corresponding to a set of $\mathsf{d}$ symbols $K$ were used to encrypt a ciphertext. That is, $c \in s(m)$ will reveal $\{\mathsf{e}_{pk} | \mathsf{e}_{pk} \in \bar{s} \wedge \mathsf{d}_{pk} \in K\}$. "Last encryption revealing" cryptosystems would, as expected, reveal the last (left-most) $\mathsf{e}_{pk}$ symbol in $\bar{s}$. Various "sequence revealing" properties may reveal the string $\bar{s}$, the string $\bar{s}$ given an unordered set of symbols in $s$, or the string $s$ itself.

Historical revelation will be defined by a parameterized function $h : \Gamma^* \times \Gamma^* \rightarrow \Sigma^*$ that takes as input a string $s$ and some auxiliary string of symbols $x$. The essential idea is that a cryptosystem will be consider *h-revealing* if there exists some probabilistic, polynomial-time algorithm $\mathcal{A}$ that is able to compute $h(s, x)$ given $x$, a message $m$, and a ciphertext $c \in s(m)$. In other words, a cryptosystem is $h$-revealing if $x$ and a ciphertext $c$, where $c \in C_s$, reveals $h(s, x)$

The auxiliary "advice" string $x$ represents *a priori* knowledge that an adversary may have. For instance, $x$ may contain public keys or keys that an adversary generated herself. An alternative formulation of $h$-revelation might do away with $x$ altogether. In fact, many of the interesting historical revelation properties that will be discussed do not make use of any $x$ advice.

**Definition 8 ($h$-Revealing Cryptosystems)** *A cascadable cryptosystem $\mathcal{C}$ is con-*

*sidered h-revealing if for a security parameter $\kappa$:*

$$\exists \mathcal{A}_{ppt},\ \exists \text{poly}(\kappa),\ \forall \rho \leftarrow \mathsf{SysGen}(1^\kappa),\ \mathcal{A}(\rho) \rightarrow (m, x);$$
$$\forall s \in \Gamma^{\text{poly}(\kappa)},\ \forall c \in s(m),\ \Pr\left[\mathcal{A}(\rho, c, m, x) = h(s, x)\right] \geq \frac{1}{\text{poly}(\kappa)}$$

One may also consider a *strongly h-revealing* cryptosystem, which is defined identically, except that an adversary will have better than a $1 - \text{negl}(\kappa)$ advantage in computing $h(s, x)$. For example, the net encryption height $\nu(s)$ of a string $s$ will often be a strongly revealed property.

By varying $h$ functions and defining different types of $x$ advice values, one may describe arbitrary historical revelation properties. Table 3.1 informally defines several potentially important or interesting revelation properties that a cascadable cryptosystem $\mathcal{C}$ might exhibit. Each row contains a name of a revelation property, a description of an advice value, and a definition of a history $h$ function.

| **Revelation Property** | Advice $x$ | History Function $h(s, x)$ |
|---|---|---|
| Height Revealing | $\epsilon$ | $\nu(s)$ |
| Known Secret Key Set Revealing | Set of $\mathsf{d}_{pk}$ symbols | $\{\mathsf{e}_{pk} \vert \mathsf{e}_{pk} \in \bar{s} \wedge \mathsf{d}_{pk} \in x\}$ |
| Known Public Key Set Revealing | Set of $\mathsf{e}_{pk}$ symbols | $\{\mathsf{e}_{pk} \vert \mathsf{e}_{pk} \in \bar{s} \wedge \mathsf{e}_{pk} \in x\}$ |
| Public Key Set Revealing | $\epsilon$ | $\{\mathsf{e}_k \vert \mathsf{e}_k \in \bar{s}\}$ |
| Last Encryption Revealing | $\epsilon$ | $\mathsf{e}_k$ if $\bar{s} = \mathsf{e}_k s'$ |
| Canonical Sequence Revealing | $\epsilon$ | $\bar{s}$ |
| Known Set, Sequence Revealing | $\{\gamma \vert \gamma \in \bar{s}\}$ | $s$ |
| Sequence Revealing | $\epsilon$ | $s$ |

Table 3.1: Example historical revelation properties

Section 3.5.5 will compare various historical revelation properties shown by constructions presented in this work. Several basic observations may be made about the interplay between cascadable semantic security, historical revelation properties, and traditional semantic security. For instance, a cascadable cryptosystem that is public-key set revealing will not be semantically secure (in the traditional sense) over domains encrypted by different public keys:

**Theorem 5** *If $\mathcal{C}$ is (known) public-key set revealing and $(\exists \mathsf{e}_{pk})$ such that $(\mathsf{e}_{pk} \in \bar{s} \wedge \mathsf{e}_{pk} \notin \bar{t})$, then $\mathcal{C}$ is not semantically secure over domain $C_s \cup C_t$.*

**Proof:** If $\mathcal{C}$ is (known) public-key set revealing, then a ciphertext $c \in s(m)$ or $c \in t(m)$ will reveal whether $\mathsf{e}_{pk}$ was used to produce $c$. An adversary in an IND-CPA experiment would be able to select $m_0 \in s(m)$ and $m_1 \in t(m)$ and could distinguish the resulting encryption by the public-key set revelation property. □

This work will not extensively delve into the issue of historical security. However, it does raise many questions. Which historical revelation properties are theoretically interesting or relevant to practice? How does historical security relate to the underlying rewrite system?

Traditional semantic security settings typically assume messages have equal length, so their encryptions are not trivially distinguishable and cascadable semantic security artificially forces messages to have the same history. Are there more natural IND-CPA experiment definitions that accommodate messages with different histories? These questions are left as open.

### 3.3.4 Cascadable CCA Security

Defining notions of security for cascadable cryptosystems stronger than the proposed cascadable semantic security is left as an open problem. Defining a suitable cascadable indistinguishability under adaptive chosen ciphertext (IND-CCA) security experiment is not as clear-cut as defining a CIND-CPA experiment. A problem arises if a particular cascadable cryptosystem's rewrite model $\mathcal{S}$ necessarily implies ciphertext malleability.

In a traditional IND-CCA experiment, an adversary has access to an $\mathcal{O}^{\mathsf{d}_{pk}}(\cdot)$ decryption oracle and may ask for decryptions of any ciphertext *except* a specific challenge $c \in \mathsf{e}_{pk}(m_b)$. An adversary in a cascadable cryptosystem may apply cascaded operations to $c$ to obtain some $c'$, then ask $\mathcal{O}^{\mathsf{d}_{pk}}(\cdot)$ to decrypt $c'$. Depending on the rules in the rewrite model $\mathcal{S}$, this may help an adversary.

For example, consider commutative rewrite models that allows decryption operations in any order, as would a cryptosystem used in a three-pass key exchange protocol. This type of rewrite model will be presented in Section 3.4.3. Suppose an adversary obtains a challenge ciphertext $c \in \mathsf{e}_{pk}(m_b)$ then computes $c' \in \mathsf{e}_A(c) = \mathsf{e}_A\mathsf{e}_{pk}(m_b)$, where $A$ is some key the adversary generated on its own. This adversary may now query $\mathcal{O}^{\mathsf{d}_{pk}}(c')$ to obtain $c'' \in \mathsf{e}_A(m_b)$, which it may trivially decrypt. Thus commutativity seems inherently incompatible with the traditional CCA setting.

A similar issue arises in re-randomizable cryptosystems where an adversary can trivially re-randomize a ciphertext and ask $\mathcal{O}^{\mathsf{d}_{pk}}(\cdot)$ for its decryption. This motivated the development of Re-playable Chosen Ciphertext (RCCA) security by Canetti, Krawczyk, and Nielson [21].

Essentially, the RCCA definition allows an adversary in a CCA experiment to ask for a decryption of any ciphertext that is not a trivial re-randomization of the challenge ciphertext. RCCA security does not immediately accommodate commutativity, since it does not restrict asking a decryption oracle to decrypt a commutatively encrypted ciphertext. Regardless, RCCA may be a useful starting point in defining a notion cascadable or commutative CCA security.

One related work is due to Dodis and Katz, who show how to construct a generic multiple encryption system that offers CCA security [36]. However, in this construction, decryption is an all-or-nothing operation – all keys must be used together to decrypt a ciphertext. One cannot remove a single layer of encryption, or for that matter, add a layer of encryption.

## 3.4 Constructions

As an exercise, Section 3.4.1 defines a trivial XOR cascadable cryptosystem that is not cascadable semantically secure. Section 3.4.2 shows how to construct a cascadable semantically secure cryptosystem $\mathcal{C}_{casc}$ from an arbitrary semantically secure cryptosystem.

Section 3.4.4 presents a commutative cascadable semantically secure cryptosystem built from an arbitrary multiplicative homomorphic, semantically secure cryptosystem (like ElGamal). The research in this chapter was originally motivated by the desire to define security for commutative cryptosystems like this one.

### 3.4.1 XOR Commutative Encryption

As an exercise, simple XOR encryption is framed in the cascadable cryptosystem framework. A commutative SRS $\mathcal{S}_{xor}$ is defined for keyspace $K = \{0,1\}^\kappa$ as follows:

$$\Gamma = \bigcup_{k \in K} \{\mathsf{e}_k\}, R = \bigcup_{k \in K} \{\mathsf{e}_k\mathsf{e}_k \rightarrow \epsilon, \bigcup_{j \in K | j < k} \mathsf{e}_j\mathsf{e}_k \rightarrow \mathsf{e}_k\mathsf{e}_j\}$$

By inspection, this SRS is canonical. Given a string $s$, its canonical form will contain all unmatched $\mathsf{e}_k$ symbols in order of key value. A cascadable cryptosystem $\mathcal{C}_{xor} = (\mathsf{SysGen}, \mathsf{G}, \mathsf{E}, \mathsf{D}, \mathcal{S}_{xor})$ is defined as:

- $\mathsf{SysGen}(1^\kappa) = 1^\kappa$
- $\mathsf{G}(1^\kappa) \rightarrow s \in \{0,1\}^\kappa$
- $\mathsf{E}(s, m) = s \oplus m$, where $m \in \{0,1\}^\kappa$
- $\mathsf{D}(s, c) = s \oplus c$

Obviously, $\mathcal{C}_{xor}$ is not cascadable semantically secure, and in fact, cannot even be framed in a CIND-CPA experiment. However, an interesting aspect is that the information theoretic security of $\mathcal{C}_{xor}$ implies it has perfect historical security – that is, no historical properties can be revealed.

A ciphertext $c$ may be the result of evaluating any $s \in \{0,1\}^\kappa$ on some message $m$. Therefore, a ciphertext $c$ yields no information on $s$. A consequence is that $\mathcal{C}_{xor}$ does not reveal any historical properties about $s$.

This illustrates how cascadable semantic security and historical security are orthogonal issues. A cryptosystem may completely reveal its entire history and still be cascadable semantically secure. Or, as is the case with $\mathcal{C}_{xor}$, a cryptosystem may not be cascadable semantically secure, yet can offer complete historical security.

### 3.4.2 A Cascadable Semantically Secure Cryptosystem

It is simple to construct in a straightforward manner, as we show in this section, a cascadable semantically secure cryptosystem from an arbitrary semantically secure cryptosystem $(\mathsf{SysGen}, \mathsf{G}, \mathsf{E}', \mathsf{D}')$. This section offers a construction $\mathcal{C}_{casc}$ as a

"proof of concept" construction in the cascadable semantic security model. The idea behind $\mathcal{C}_{casc}$ is simple: encrypt a ciphertext, encode it as a list of plaintexts, and encrypt each element.

$\mathcal{C}_{casc}$ will consist of four probabilistic, polynomial-time algorithms $\mathsf{SysGen}, \mathsf{G}, \mathsf{E}, \mathsf{D}$ and the rewrite model $\mathcal{S}_{casc}$ defined in definition 2. Let message space $M = \Sigma^{\ell(\kappa)}$, for some fixed function $\ell$. Let $\mathsf{SysGen}(1^\kappa) \rightarrow \rho$, $\mathsf{G}(\rho) \rightarrow (pk, sk)$ and $\mathsf{E}'$ and $\mathsf{D}'$ have the properties that $\forall (pk, sk) \in G(\rho)$, $\forall m \in \Sigma^{\ell(\kappa)}$, $\mathsf{D}'(sk, \mathsf{E}'(pk, m)) = m$. Let $\xi : \Sigma^* \rightarrow \Sigma^*$ be an efficiently computable, efficiently invertible function that outputs a prefix-free encoding of an arbitrary input.

$\mathcal{C}_{casc}$ is defined in figures 3-2 and 3-3.

CASCADABLE ENCRYPTION ON INPUTS $pk$ AND $x \in \Sigma^*$:

1. Let $m = \xi(x)||\xi(pk)$, where $||$ signifies concatenation.
2. Parse $m$ into blocks in $\Sigma^{\ell(\kappa)}$ denoted $(m_1, \ldots, m_n)$, padding as necessary.
3. Define $\mathsf{E}_{pk}(x) = \xi(\mathsf{E}'_{pk}(m_1))|| \ldots ||\xi(\mathsf{E}'_{pk}(m_n))$.

Figure 3-2: Cascadable encryption from a generic, semantically secure cryptosystem.

CASCADABLE DECRYPTION ON INPUT $sk$ AND $x \in \Sigma^*$:

1. Attempt decode $x$ under $\xi^{-1}$ as elements $c_i = \xi^{-1}(x_i)$.
2. Output $\perp$ if decoding fails.
3. Attempt to decrypt each $c_i$ as $m_i = \mathsf{D}'_{sk}(c_i)$.
4. If any $\mathsf{D}'_{sk}$ operation fails or some $c_i$ is outside the domain of $\mathsf{D}'$, output $\perp$.
5. Attempt to decode $m_1 m_2 \ldots m_n$ under $\xi^{-1}$ as $x||pk$.
6. If $pk$ is not the public key associated with $sk$ or decoding fails, output $\perp$.
7. Otherwise output $x$.

Figure 3-3: Cascadable decryption from a generic, semantically secure cryptosystem.

**Theorem 6** *$\mathcal{C}_{casc}$ exhibits rewrite fidelity.*

**Proof:** To prove its security, it is important to first show that $\mathcal{C}_{casc}$ exhibits rewrite fidelity, that is $\forall s \in \Gamma^*$, $\forall m \in M$, $s(m) = \bar{s}(m)$. Consider a pair of values $t$ and $t'$ along the derivation path from $s$ to $\bar{s}$ such that $t'$ is derivable by applying a single

rule to $t$: $s \xrightarrow{*} t \xrightarrow{1} t' \xrightarrow{*} \bar{s}$. In $\mathcal{C}_{casc}$'s rewrite model $\mathcal{S}_{casc}$ specified in definition 2 on page 50, there is only one type of rule $\mathsf{d}_{pk}\mathsf{e}_{pk} \rightarrow \epsilon$.

Thus, it must be the case that $t = u\mathsf{d}_{pk}\mathsf{e}_{pk}v$ and $t' = uv$. Clearly, it will be the case that $t(m) = t'(m)$ if $\mathsf{d}_{pk}\mathsf{e}_{pk}v(m) = v(m)$. Consider an element $c \in v(m)$. If $c = \perp$, then clearly $\mathsf{d}_{pk}\mathsf{e}_{pk}\perp = \perp$.

Otherwise, when evaluating $\mathsf{e}_{pk}$, $\mathsf{E}$ will encode $v(m)||pk$ under $\xi$ into blocks $m_1, \ldots, m_j$ from $\Sigma^{\ell(\kappa)}$, and encrypt and encode the blocks as $\xi(\mathsf{E}(pk, m_1))|| \ldots ||\xi(\mathsf{E}(pk, m_j))$. Evaluating the subsequent $\mathsf{d}_{pk}$ symbol with $\mathsf{D}$ will simply decode and decrypt each ciphertext under $sk$. Since $\forall x \in \Sigma^*$, $(\xi^{-1}(\xi(x)) = x)$ and $\forall (pk, sk) \in G(\rho)$, $\forall m \in M$, $(\mathsf{D}'(sk, \mathsf{E}'(pk, m)) = m)$, the cascaded decryption specified in figure 3-3 will faithfully recover each $m_i$, obtain $v(m)||pk$, and output some $c \in v(m)$.

Thus it is the case that $\mathsf{d}_{pk}\mathsf{e}_{pk}c = c$ for any $c \in v(m)$. Thus, $\forall m \in M$, $t(m) = t'(m)$. Since $t$ was chosen arbitrarily along the derivation path from $s$ to $\bar{s}$ and $s$ is an arbitrary term, we can inductively argue that $\forall s \in \Gamma^*$, $\forall m \in M$, $s(m) = \bar{s}(m)$. In other words, $\mathcal{C}_{casc}$ exhibits rewrite fidelity. □

**Theorem 7** *$\mathcal{C}_{casc}$ is cascadable semantically secure.*

**Proof:** Assume for the sake of contradiction that there is some probabilistic polynomial time adversary $c\mathsf{Adv}$ that has greater than a polynomial advantage $1/\text{poly}(\kappa)$ in the experiment $\text{Exp}^{CIND-CPA}_{\mathcal{C}_{casc}, c\mathsf{Adv}}[\kappa, n]$, for some $n$ that is polynomial in $\kappa$.

The traditional IND-CPA experiment will be reduced to the CIND-CPA experiment by constructing a probabilistic polynomial time adversary $\mathsf{Adv}$ that makes calls to $c\mathsf{Adv}$. $\mathsf{Adv}$ will contradict the assumed semantic security of the generic cryptosystem $(\mathsf{G}, \mathsf{E}, \mathsf{D})$.

Let $(pk^*, sk^*)$ be the key pair output by $\mathsf{G}(\rho)$ in the traditional IND-CPA experiment, as stated in definition 6. $\mathsf{Adv}$ obtains public $pk^*$ and system parameters $\rho$ in its standard IND-CPA experiment. $\mathsf{Adv}$ then generates $(n-1)$ other $(pk, sk)$ pairs, and passes $\rho$ and all $n$ public keys in random order to $c\mathsf{Adv}$. These values are distribution identically as the input that $c\mathsf{Adv}$ expects in the CIND-CPA experiment.

$c\mathsf{Adv}$ then outputs $m_0, m_1 \in M$ and $s \in \Gamma^*$. Because $\mathcal{C}_{casc}$ exhibits rewrite fidelity, it is the case that $\forall m$, $s(m) = \bar{s}(m)$. Furthermore, by the definition of the cascading CIND-CPA experiment, $\bar{s}$ must contain an encryption symbol $\mathsf{e}_{pk}$ corresponding to at least one $pk$ initially input to $c\mathsf{Adv}$. Since public keys are randomly generated and permuted when input to $c\mathsf{Adv}$, there is at least a $1/n$ chance that the first symbol of $\bar{s}$ (i.e. the last operation) is $\mathsf{e}_{pk^*}$. Assume this to be the case and that $\bar{s} = \mathsf{e}_{pk^*}s'$.

Assume $\mathsf{Adv}$ obtains all key material generated by $c\mathsf{Adv}$. This allows $\mathsf{Adv}$ to evaluate any symbol output by $c\mathsf{Adv}$ *except* $\mathsf{d}_{pk^*}$. Because of rewrite fidelity, $\mathsf{Adv}$ can evaluate $\bar{s}$ without changing $c\mathsf{Adv}$'s input distribution. Since $\nu(s) > 0$, by the model $\mathcal{S}$, $\bar{s}$ cannot contain any decryption symbols. Thus, $\mathsf{Adv}$ can perfectly simulate $c\mathsf{Adv}$'s expected input $s(m_b)$.

Since it was assumed that $\bar{s} = \mathsf{e}_{pk^*}s'$, $c\mathsf{Adv}$ will expect input from $\mathsf{e}_{pk^*}(s'(m_b))$. Consider the last step of this operation: the cascaded encryption defined in figure 3-2 would parse $c \in s'(m_b)$ into $\ell(\kappa)$-bit blocks denoted $m_1^b, \ldots, m_j^b$. Given

$\mathsf{E}(pk^*, m_1^b)\| \ldots \|\mathsf{E}(pk^*, m_j^b)$, $\mathsf{Adv}$ will correctly identify $b$ with non-negligible probability.

Consider "hybrid" cascadable ciphertexts of the form:

$$c_i = \mathsf{E}(pk^*, m_1^0)\| \ldots \|\mathsf{E}(pk^*, m_{i-1}^0)\|\mathsf{E}(pk^*, m_i^1)\| \ldots \|\mathsf{E}(pk^*, m_j^1)$$

In other words, the first $(i-1)$ blocks of a hybrid ciphertext will be spliced in from an encryption of $m^0$, while the remaining $(j-i)$ blocks will be splied from an encryption of $m^1$. Thus, there will be $j$ different hybrid ciphertexts indexed from 1 to $j$. Hybrid ciphertext $i$ and $(i+1)$ will differ in the $i$th block.

By a standard hybrid argument, there must exist two hybrid ciphertexts $c_i$ and $c_{i+1}$ where $c\mathsf{Adv}$ maintains at least a $1/(j \cdot \text{poly}(\kappa))$ advantage. (More on the bounds of $j$ later.) $\mathsf{Adv}$ may take advantage of this for its own IND-CPA experiment. To do so, it will output the messages $m_i^0$ and $m_i^1$, and will obtain $\mathsf{E}(pk^*, m_i^b)$ in response. $\mathsf{Adv}$ will the provide the following ciphertext to $c\mathsf{Adv}$:

$$\mathsf{E}(pk^*, m_1^0)\| \ldots \|\mathsf{E}(pk^*, m_{i-1}^0)\|\mathsf{E}(pk^*, m_i^b)\|\mathsf{E}(pk^*, m_{i+1}^1)\| \ldots \|\mathsf{E}(pk^*, m_j^1)$$

This ciphertext will be one of the two hybrids that $c\mathsf{Adv}$ maintains a polynomial advantage in distinguishing. Thus, considering there is a $1/n$ chance that the first symbol of $s$ is $\mathsf{e}_{pk^*}$, $\mathsf{Adv}$ has a $1/(j \cdot n \cdot \text{poly}(\kappa))$ advantage in its own IND-CPA experiment.

One outstanding issue is the value of $j$. If $\mathcal{C}_{casc}$ is *not* efficiently cascadable, then it is possible that $j$ is exponential in $\kappa$. However, $c\mathsf{Adv}$ must still run in time polynomial in $\kappa$. If provided $c \in s(m_b)$ on a tape, it can only access some polynomial number $h(\kappa)$ bit of $c$. Thus, $j$ is effectively upper bounded by the number of *accesses* that $c\mathsf{Adv}$ makes to its challenge ciphertext. $\mathsf{Adv}$ can generate its hybrid ciphertext by providing $\mathsf{E}(pk^*, m^0)$ blocks on the first $i-1$ queries, then $\mathsf{E}(pk^*, m_i^b)$, then $\mathsf{E}(pk^*, m^1)$ blocks on all remaining queries.

Thus, $\mathsf{Adv}$ will have at least a $1/(h(\kappa) \cdot n \cdot \text{poly}(\kappa))$ advantage in distinguishing $\mathsf{E}(pk^*, m_i^b)$ ciphertexts through calls to $c\mathsf{Adv}$. Since this advantage is polynomial in $\kappa$, it contradicts the assumed semantic security of the underlying $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ cryptosystem. Therefore, no such polynomial time $c\mathsf{Adv}$ can exist and $\mathcal{C}_{casc}$ must be cascadable semantically secure. □

**$\mathcal{C}_{casc}$ historical revelation properties.** The underlying encryption scheme $\mathsf{E}'$ may reveal various historical properties. For example, if the length of the encoding $\xi$ and encryption $\mathsf{E}'$ outputs are predictable, then $\mathcal{C}_{casc}$ would be *height revealing*. If $\mathsf{E}'$ included the public key with a ciphertext output, then $\mathcal{C}_{casc}$ would be *last encryption revealing*.

### 3.4.3 A Commutative String Rewrite System Model

We now define a simple, canonical SRS that models commutative encryption:

**Definition 9 (Commutative Cryptosystem Model)** *Let $K$ be a keyspace and $\epsilon$ represent an empty string. Define the string rewrite system $\mathcal{S}_{com} = (\Gamma, R)$ as follows:*

$$\Gamma = \bigcup_{k \in K} \{\mathsf{e}_k, \mathsf{d}_k\}, \quad R = \bigcup_{k \in K} \left\{ \mathsf{d}_k \mathsf{e}_k \rightarrow \epsilon, \bigcup_{j \in K - \{k\}} \mathsf{d}_k \mathsf{e}_j \rightarrow \mathsf{e}_j \mathsf{d}_k \right\}$$

The alphabet $\Gamma$ contains symbols for encryption or decryption under all possible keys. There are two types of rules in $\mathcal{S}_{com}$. The first are familiar decryption rules of the form $\mathsf{d}_k \mathsf{e}_k \rightarrow \epsilon$, which are identical to those in the basic cascading model $\mathcal{S}_{casc}$ from definition 2. The second type of rule, of the form $\mathsf{d}_k \mathsf{e}_j \rightarrow \mathsf{e}_j \mathsf{d}_k$, is less intuitive. These rules allow a decryption symbol $\mathsf{d}_k$ to skip past any encryption under a different key $\mathsf{e}_j$. For this reason, these rules are informally referred to as "$\mathsf{d}$-skip" rules.

The intuition behind $\mathsf{d}$-skip rules are that one should be able to decrypt any existing encryptions. Shamir identified this relationship in the "third commutative diagram" in [105]. This is the commutative property used in most applications of commutativity [1, 27, 106]. A decryption symbol may skip past the immediately preceding encryption symbol, if that encryption is under a different key. One may think of the decryption symbol $\mathsf{d}_i$ as being able to scan from left to right along a string until it matches a corresponding encryption $\mathsf{e}_i$ or can no longer move right. For example, consider the string $s = \mathsf{d}_k \mathsf{e}_j \mathsf{e}_k$. In $\mathcal{S}_{casc}$, we'd have that $s$ would be in its canonical form since no rules could be applied. However, in $\mathcal{S}_{com}$, one may apply the $\mathsf{d}$-skip rule $\mathsf{d}_k \mathsf{e}_j \rightarrow \mathsf{e}_j \mathsf{d}_k$, followed by a decryption rule $\mathsf{d}_k \mathsf{e}_k \rightarrow \epsilon$ to obtain the canonical form $\bar{s} = \mathsf{e}_j$.

Why use $\mathsf{d}$-skip rules instead of something more intuitive, like $\mathsf{e}_j \mathsf{e}_k \leftrightarrow \mathsf{e}_k \mathsf{e}_j$? Unfortunately, such rules would create derivation cycles and $\mathcal{S}_{com}$ would neither be terminating nor canonical. The definition of equivalence requires a canonical rewrite system, so non-terminating rewrite systems cannot be used in the definition of cascadable semantic security.

Suppose a more intuitive rewrite system, denoted $\mathcal{S}'_{com}$, used rules of the form $\mathsf{e}_j \mathsf{e}_k \leftrightarrow \mathsf{e}_k \mathsf{e}_j$ instead of $\mathsf{d}$-skip rules. Although $\mathcal{S}'_{com}$ is not canonical, one could adapt it for use in cascadable cryptosystems by defining a "class rewrite system" [32, 33, 34]. Given a string $s$, define its class $C(s) = \{t \in \Gamma^* | s \overset{*}{\leftrightarrow} t\}$. Two strings $s$ and $t$ will be considered equivalent if $C(s) = C(t)$. A string $\mathsf{e}_x \mathsf{e}_y \mathsf{e}_z$ in $\mathcal{S}'_{com}$ shares a class with every permutation of $\{\mathsf{e}_x, \mathsf{e}_y, \mathsf{e}_z\}$.

For simplicity, this work will stick to basic, terminating string rewrite systems. It is left as an open problem to define a notion of security that is compatible with non-terminating rewrite systems, and for that matter, general term rewrite systems.

**Theorem 8** *$\mathcal{S}_{com}$ is canonical.*

**Proof: $\mathcal{S}$ is terminating.** Suppose there is a derivation cycle in $\mathcal{S}$. Let $s$ be a string in the cycle, and $(r_1, \ldots, r_n)$ be the minimum length sequence of rules such that $s \rightarrow_{r_1} s_1 \rightarrow_{r_2} \ldots s_{n-1} \rightarrow_{r_n} s$. Clearly, since there are no length-increasing rules in $\mathcal{S}$, no rule $r_i$ can be a decryption rule. Otherwise, by repeating a derivation cycle, a string would dwindle to an empty string.

Therefore, each $r_i$ must be a $\mathsf{d}$-skip rule. The position of some $\mathsf{d}$ symbol must decrease and move to the right in each subsequent derivation $s_i$. Since there is no rule that can move any $d$ symbol to the left, there is no way to return to the original string $s$. For example, once a string $\mathsf{d}_i\mathsf{e}_j$ is replaced with $\mathsf{e}_j\mathsf{d}_i$, there is no way to return back to the original string. Thus, there cannot be any derivation cycles in $\mathcal{S}$, and so it is terminating.

**$\mathcal{S}$ is confluent.** Recall that confluence means that for all $s \in \Gamma^*$ if $s \xrightarrow{*} t$ and $s \xrightarrow{*} u$, there exists some $w$ such that $t \xrightarrow{*} w$ and $u \xrightarrow{*} w$. A weaker property is *local confluence*, which is the property that $s \in \Gamma^*$ if $s \to t$ and $s \to u$, then there exists some $w$ such that $t \xrightarrow{*} w$ and $u \xrightarrow{*} w$. In other words, any single-rule descendants of $s$ must have a common descendant.

A well known result is that a terminating, locally confluent system is confluent [33]. Thus, it will be sufficient to show that $\mathcal{S}$ is locally confluent. Consider a string $s$ that derives two strings $t$ and $u$ under respective rules $r$ and $r'$.

In $\mathcal{S}$, $r$ and $r'$ are only applied to length-2 sub-strings of $s$. Recall that all of the rewritten strings of rules in $\mathcal{S}$ are either of the form $\mathsf{d}_i\mathsf{e}_i$ or $\mathsf{d}_i\mathsf{e}_j$. Wherever $r$ and $r'$ are applied in $s$, they cannot overlap. So, every $r$ and $r'$ must be applied to disjoint sub-terms. Then it is trivial to see that applying $r'$ to $t$ yields the same string as applying $r$ to $u$.

Since every $t$ and $u$ derived in one step from $s$ have a common descendant, $\mathcal{S}$ is locally confluent. Since it is also terminating, it is confluent. Thus, $\mathcal{S}$ is canonical. □

### 3.4.4 Commutativity from Homomorphic Encryption

We show how to build a commutative, cascadable semantically secure cryptosystem from an arbitrary, semantically secure cryptosystem that supports homomorphic multiplication of ciphertexts, like ElGamal. This is the fundamental idea behind the detailed construction presented in Sections 3.5.1 and 3.5.2.

**Theorem 9** *A semantically secure, re-randomizable cryptosystem* $(\mathsf{SysGen}, G, E', D')$ *that supports homomorphic multiplication of ciphertexts implies the existence of a commutative, cascadable semantically secure cryptosystem* $\mathcal{C} = (G, E, D, \mathcal{S}_{com})$.

**Proof:** This section describes an abstract construction of a commutative semantically secure cryptosystem. Section 3.5 contains a detailed construction based on ElGamal that is proved to be cascadable semantically secure in Section 3.5.4.

The function $\mathsf{E}'$ supports homomorphic multiplication of ciphertexts, that is, given $\mathsf{E}'_{pk}(m_1)$ and $\mathsf{E}'_{pk}(m_2)$, one may compute $\mathsf{E}'_{pk}(m_1 \cdot m_2)$ as $\mathsf{E}'_{pk}(m_1) \cdot \mathsf{E}'_{pk}(m_2)$. Alternatively, $\mathsf{E}'$ only needs to support homomorphic multiplication of a ciphertext by some constant, that is, given $\mathsf{E}'_{pk}(m)$ and $x$, one can compute $\mathsf{E}'_{pk}(xm)$.

Ciphertexts in the cryptosystem $(\mathsf{G}, \mathsf{E}, \mathsf{D}, \mathcal{S}_{com})$ will be a list of tagged $\mathsf{E}'$ ciphertexts, each of the form $(pk, \mathsf{E}'(pk, v))$. Denote the space of lists of tagged ciphertexts of length $i$ as $C_i$. Define commutative encryption as shown in figure 3-4.

COMMUTATIVE ENCRYPTION $\mathsf{E}(pk, z)$:

1. Input is public key $pk$, message $z \in C_n$ for $n \geq 0$, and randomness.
2. If the input is malformed, return $\perp$.
3. If $n = 0$, return $(pk, \mathsf{E}'(pk, z))$.
4. Denote $z = [(pk_1, \mathsf{E}'(pk_1, v_1)), \ldots, (pk_n, \mathsf{E}'(pk_n, v_n))]$.
5. Generate $(n+1)$ random values $r_i$ such that $\prod_{i=1}^{n+1} r_i = 1$.
6. Homomorphic multiply to compute $c'_i = \mathsf{E}'(pk_i, v_i \cdot r_i)$ for $i = 1, 2, \ldots, n$.
7. Re-randomize each $c'_i$.
8. Let $c'_{n+1} = \mathsf{E}'(pk, r_{n+1})$.
9. Return $(pk_1, c'_1), \ldots, (pk_{n+1}, c'_{n+1})$.

Figure 3-4: Commutative encryption from multiplicative homomorphic encryption

When given a plaintext message $m$, $\mathsf{E}$ simply returns the result of the underlying $\mathsf{E}'$ encryption function, tagged with the public key used to encrypt it. Otherwise, $\mathsf{E}$'s input will be a list of $n$ tagged ciphertexts output from $\mathsf{E}'$. These ciphertexts have the property that *the product of their decryptions is the plaintext*. In other words, $\prod_{i=1}^{n} \mathsf{D}(sk_i, c_i) = m$. $\mathsf{E}$ generates a new ciphertext that maintains this property. To do so, $\mathsf{E}$ selects $(n+1)$ random $r_i$ values such that $\prod_{i=1}^{n+1} r_i = 1$. $\mathsf{E}$ homomorphically multiplies an $r_i$ value into each $\mathsf{E}'$ ciphertext, then encrypts the final randomization $r_{n+1}$ as $\mathsf{E}'(pk, r_{n+1})$, and finally outputs all $(n+1)$ tagged ciphertexts. These operations preserve that $\prod_{i=1}^{n+1} \mathsf{D}(sk_i, c_i) = m$.

The process for decrypting commutative ciphertexts is illustrated in figure 3-5.

Decryption is fairly straightforward. Someone with a secret key $sk$ will first scan through a list of $n$ ciphertexts from $\mathsf{E}'$ and check for any that have been encrypted with a corresponding public key $pk$. If there are multiple such values, $\mathsf{D}$ can pick one arbitrarily and decrypt it to obtain some value $\mathsf{D}'(sk, c_k) = v_k$. $\mathsf{D}$ will then choose $(n-1)$ values $r_i$ such that $\prod_{j \neq k} r_j = v_k$ and homomorphically multiply each $r_j$ into each corresponding $c_j$ ciphertexts. Thus, by the following, it is the case that correctness of decryption will be preserved:

$$m = \prod_{i=1}^{n} v_i = v_k \prod_{j \neq k}^{n} v_j = \prod_{j \neq k}^{n} r_j v_j$$

In practice, ElGamal would satisfy this definition. However, there are two weaknesses in this formulation related to historical security. First, is that this scheme is *public key set revealing*. Someone trivially knows which public keys have encrypted

COMMUTATIVE DECRYPTION $\mathrm{D}(sk, z)$:

1. Input is a secret key $sk$ and a ciphertext $z \in C_n$, for $n \geq 1$.
2. If the input is malformed, return $\perp$.
3. Denote $z = [(pk_1, \mathsf{E}'(pk_1, v_1), \ldots, (pk_n, \mathsf{E}'(pk_n, v_n)]$.
4. Search for an element $c_k$ such that $c_k = (pk, \mathsf{E}'(pk, v_k))$, where $pk$ is $sk$'s corresponding public key.
5. If no such element exists, return $\perp$.
6. If $n = 1$, and $z = (pk, c)$, return $m = \mathsf{D}'(sk, c)$.
7. Else, choose $(n-1)$ random $r_j$ values such that $\prod_{j \neq k}^{n} r_j = \mathsf{D}'(sk, c_k) = v_k$.
8. Let $c'_j = \mathsf{E}'(pk_j, v_j \cdot r_j)$ and re-randomize $c'_j$.
9. Return $(c'_1, \ldots, c'_{k-1}, c'_{k+1}, c'_n)$.

Figure 3-5: Commutative decryption from multiplicative homomorphic decryption

a message. It is also *canonical sequence revealing*, since it reveals the exact order of operations. Obviously, it is also *height revealing*.

Fortunately, all of these historical properties, except height, can be hidden by using a universal re-encryption scheme, such as one due to Golle et al. [53]. Universal re-encryption allows one to re-randomize without the corresponding public key, which allows us to remove $pk$ values from the ciphertext. One may also simply permute the order of the ciphertext blocks after each operation, which removes any sequence information. A detailed description of just such a cryptosystem, denoted $\mathcal{C}_{com}$ is given in Section 3.5.

### 3.4.5 Commutativity from Semantic Security

We will briefly digress to sketch a commutative cryptosystem based on a generic semantically secure cryptosystem $(\mathsf{G}', \mathsf{E}', \mathsf{D}')$ and simple XOR operations. This construction will demonstrate that the cascadable semantic security definition given in section 3.3.2 is not sufficient to guarantee a reasonable notion of security for some protocols, like three-pass key exchange [105]. This discrepancy motivates the development of a new notion of *transcript security*, which is informally defined below in definition 11.

The construction is quite simple. Assume $(\mathsf{G}', \mathsf{E}', \mathsf{D}')$ is a standard semantically secure, public-key cryptosystem. Each cascaded encryption will generate a random value $r$, then XOR $r$ with the message payload as $m \oplus r$, and finally append a labeled public-key encryption $(pk, \mathsf{E}'(pk, r))$ to the output:

XOR Commutative Encryption $\mathrm{E}(pk, z)$:

1. Input is public key $pk$, message $z \in C_n$ for $n \geq 0$, and randomness.
2. If the input is malformed, return $\perp$.
3. Denote $z = [c, (pk_1, c_1), \ldots, (pk_n, c_n)]$.
4. Generate a random value $r$ and compute $c_{n+1} = \mathsf{E}'(pk, r)$.
5. Return $z' = [c \oplus r, (pk_1, c_1), \ldots, (pk, c_{n+1})]$.

Figure 3-6: Commutative encryption from semantic security

To decrypt, a party can search for its public key among the list of ciphertexts, decrypt the specific ciphertext, and XOR the decryption out of the message payload. However, consider an application like three-pass key exchange. As can be seen in figure 3-1, an eavesdropper would obtain three ciphertexts: $[m \oplus r_a, (a, \mathsf{E}'(a, r_a))]$, $[m \oplus r_a \oplus r_b, (a, \mathsf{E}'(a, r_a)), (b, \mathsf{E}'(b, r_b))]$, and $[m \oplus r_b, (b, \mathsf{E}'(b, r_b))]$. Clearly, an eavesdropper can obtain $m$ by comparing these ciphertexts.

Yet based on the underlying semantic security of $\mathsf{E}'$, this construction appears[2] to be a cascadable, semantically secure cryptosystem. This is not a flaw or contradiction of the cascadable semantic security definition. In the CIND-CPA experiment an adversary receives a single ciphertext encrypted under an arbitrary $s$. Adversaries do not receive intermediate ciphertexts, as an eavesdropper would when monitoring a three-pass key exchange protocol.

**Transcript Security:** This discrepancy motivates a new notion of *transcript security*. The idea is that an adversary should be able to obtain a transcript of all intermediate, non-trivial ciphertexts used to generate a particular cascaded ciphertext $c \in s(m)$. The cryptosystem as described in figure 3-6 would fail to meet such a definition.

To address this issue, we introduce a CIND-CPA experiment with *transcripts*. Rather than simply obtaining an encryption $c \in s(m_b)$, as in the CIND-CPA experiment, an adversary will obtain an intermediate ciphertext for non-trivial suffixes of $s$ denoted $s_1, \ldots, s_j$. The symbolic difference between each non-trivial suffix will be denoted by $t_i$. That is, $s_i = t_i \circ s_{i-1}$ and $t_1 = s_1$. Thus, $s_i = t_i \circ s_{i-1} = t_i \circ t_{i-1} \circ \ldots \circ t_1(m)$.

Note that a traditional chosen-plaintext attack could obtain independent encryptions of any non-trivial suffix $s_i(m_b)$. However, we wish to give the adversary access to ciphertexts that *depend* on previous encryptions. Thus, we will provide the adversary $c_1 \in t_1(m_b)$ and $c_i \in t_i(c_{i-1})$ for each $1 < i \leq j$.

[2]This observation is upon inspection and no formal proof is provided.

**Definition 10 (CIND-CPAT Experiment)** *A cascadable indistinguishability under chosen plaintext attack with transcripts experiment with adversary* $\mathsf{Adv}$ *and cascadable cryptosystem* $\mathcal{C}$ *is defined as follows:*

$\text{Exp}^{CIND-CPA}_{\mathsf{Adv},\mathcal{C}}[\kappa, n]$*:*

1. $\mathsf{SysGen}(1^\kappa) \rightarrow \rho$
2. *Call* $\mathsf{G}(\rho)$ $n$ *times to generate* $(pk_1, sk_1), \ldots, (pk_n, sk_n)$
3. *Denote the key symbol set* $K = \{\mathsf{e}_{pk_1}, \ldots, \mathsf{e}_{pk_n}\}$.
4. $\mathsf{Adv}(\rho, pk_1, \ldots, pk_n)$ *finds* $(m_0, m_1, s)$ *such that* $m_0, m_1 \in M$.
5. *Let* $s_1, \ldots, s_j$ *be the set of suffixes of* $s$ *such that:*
   (a.) $s_j = s$
   (b.) $(\forall s_i)(\infty > \nu(s_i) > 0)$
   (c.) $(\forall s_i)(\exists \mathsf{e}_{pk})(\mathsf{e}_{pk} \in K \wedge \mathsf{e}_{pk} \in \bar{s_i})$
6. *Let* $t_1, \ldots, t_j$ *be the set of symbolic differences between* $s_i$ *suffixes such that:*
   (a.) $s_1 = t_1$
   (b.) $s_i = t_i \circ s_{i-1}$ *for* $1 < i \leq j$
7. *Let* $b \overset{R}{\leftarrow} \{0, 1\}$.
8. *Let* $c_1 \in t_1(m_b)$ *and* $c_i \in t_i(c_{i-1})$ *for* $1 < i \leq j$
9. $\mathsf{Adv}(c_1, \ldots, c_j)$ *outputs guess bit* $b'$

Cryptosystems where no polynomial time adversary can gain a significant advantage in this experiment are considered *transcript secure*:

**Definition 11 (Transcript Security)** *A cascadable cryptosystem* $\mathcal{C} = (\mathsf{SysGen}, \mathsf{G}, \mathsf{E}, \mathsf{D}, \mathcal{S})$ *is transcript secure if:*

$$\forall n = \text{poly}(\kappa),\ \forall \mathsf{Adv} \in Time(\text{poly}(\kappa))\ \Pr\left[\text{Exp}^{CIND-CPA}_{\mathsf{Adv},\mathcal{C}}[\kappa, n]\ \textit{succeeds}\ \right] \leq \frac{1}{2} + \text{negl}(\kappa)$$

Clearly, the cryptosystem described in figure 3-6 is not transcript secure. An adversary could obtain intermediate encryptions of the three non-trivial suffixes $s = s_3 = \mathsf{d}_a\mathsf{e}_b\mathsf{e}_a$, $s_2 = \mathsf{e}_b\mathsf{e}_a$, and $s_1 = \mathsf{e}_a$. The differences between these suffixes are $t_3 = \mathsf{d}_a$, $t_2 = \mathsf{e}_b$, and $t_1 = \mathsf{e}_a$. The adversary would obtain ciphertexts $c_3 = \mathsf{d}_a(c_2)$, $c_2 = \mathsf{e}_b(c_1)$, and $c_1 = \mathsf{e}_a(m_b)$, which would trivially allow it to distinguish (and extract) $m_b$.

This chapter will not elaborate further on transcript security. The author conjectures that the constructions presented in sections 3.4.4 and 3.5.1 are in fact transcript secure. Proving these cryptosystems to be transcript secure is left as an open problem.

## 3.5 A Universally Re-encryptable, Commutative Cryptosystem

We present a commutative cryptosystem that is cascadable semantically secure and reveals less information than the construction presented in Section 3.4.4.

### 3.5.1 Universal Re-Encryption based on ElGamal

Section 3.5.2 presents a practical commutative public-key cryptosystem based on Golle et al.'s universal re-encryption based on ElGamal [53], referred to as GJJS. The GJJS universally re-randomizable cryptosystem is defined as follows:

- System Parameters: $\mathsf{SysGen}(1^\kappa) \rightarrow (\mathbb{G}, g)$, where $\mathbb{G}$ is a group with prime order $q$ and $g$ is a generator of $\mathbb{G}$.
- Key Generation: $\mathsf{G}(\mathbb{G}, g) \rightarrow (y = g^x, x) = (pk, sk)$, where $x \in_R \mathbb{Z}_q$.
- Encryption: $\mathsf{uE}(\mathbb{G}, g, y, r, s, m) \rightarrow c = [(\alpha, \beta); (\gamma, \delta)] = [(my^r, g^r); (y^s, g^s)]$, where $y \in \mathbb{G}$ , $m \in \mathbb{G}$, and random $(r, s) \in \mathbb{Z}_q^2$. This may be abbreviated as $c = \mathsf{uE}_y(m)$.
- Decryption: $\mathsf{uD}(\mathbb{G}, g, x, c)$ where $x \in Z_q$ and $c = [(\alpha, \beta); (\gamma, \delta)] \in \mathbb{G}^4$: Test if $\gamma/\delta^x = 1$. If so, output $m = \alpha/\beta^x$. Otherwise output $\perp$. This may be abbreviated as $m = \mathsf{uD}_x(c)$.
- Universal Re-Encryption: $\mathsf{uRe}(c, t, u) \rightarrow [(\alpha\gamma^t, \beta\delta^t); (\gamma^u, \delta^u)]$, where $c = [(\alpha, \beta); (\gamma, \delta)] \in \mathbb{G}^4$ and random $(t, u) \in \mathbb{Z}_q^2$. Abbreviate this operation as $c' = \mathsf{uRe}(c)$.

### 3.5.2 Commutative Cryptosystem Construction

We now define a commutative cryptosystem $\mathcal{C}_{com} = (\mathsf{SysGen}, \mathsf{G}, \mathsf{E}, \mathsf{D}, \mathcal{S}_{com})$ based on the GJJS universally re-randomizable cryptosystem $(\mathsf{SysGen}, \mathsf{G}, \mathsf{uE}, \mathsf{uD})$. Figure 3-7 specifies the method for encrypting commutative ciphertexts using the GJJS cryptosystem.

When given a plaintext message $m \in \mathbb{G}$, $\mathsf{E}$ simply outputs the result of the underlying $\mathsf{uE}$ encryption function. Otherwise, $\mathsf{E}$'s input will be a list of $n$ ciphertexts output by $\mathsf{uE}$, which are each four-tuples of elements in $\mathbb{G}$. These ciphertexts have the property that *the product of their decryptions is the plaintext.* In other words, $\prod_{i=1}^{n} \alpha_i/\beta_i^{x_i} = m$. $\mathsf{E}$ generates a new ciphertext that maintains this property.

To do so, $\mathsf{E}$ selects $(n + 1)$ values $r_i$ such that $\prod_{i=1}^{n+1} r_i = 1$. $\mathsf{E}$ multiplies an $r_i$ value into each $\alpha_i$ and re-randomizes the resulting ciphertext with $\mathsf{uRe}$. Thus, this commutative cryptosystem relies on the homomorphic properties of the GJJS cryptosystem. $\mathsf{E}$ then then encrypts the final randomization $r_{n+1}$ as $\mathsf{uE}_y(r_{n+1})$ and permutes the entire list of $(n+1)$ $\mathsf{uE}$ ciphertexts. These operations preserve the property that $\prod_{i=1}^{n+1}(r_i\alpha_i)/\beta_i^{x_i} = m$.

Figure 3-8 specifies the method for decrypting commutative ciphertexts.

Again, decryption is fairly straight forward. Someone with a secret key $x$ will first scan through a list of $n$ $\mathsf{uE}$ ciphertexts and check for any that have been encrypted with a corresponding public key $y$. To do so, $\mathsf{D}$ tests whether any $\gamma_k/\delta_k^x = 1$. If there are multiple such values, $\mathsf{D}$ can pick one arbitrarily and decrypt it to obtain some value $\alpha_k/\beta_k^x$. $\mathsf{D}$ will then choose $(n-1)$ values $r_i$ such that $\prod_{j\neq k} r_j = \alpha_k/\beta_k^x$, multiply

COMMUTATIVE ENCRYPTION $\mathrm{E}(y, z)$:

1. Input is public key $y \in \mathbb{G}$, message $z \in \mathbb{G}$ or $z \in \mathbb{G}^{4n}$ for $n \geq 1$, and randomness.
2. Otherwise, return $\perp$.
3. If $z \in \mathbb{G}$ return $\mathsf{uE}_y(z)$.
4. Else, parse $z$ as a list of $n$ elements of the form $c_i = [(\alpha_i, \beta_i); (\gamma_i, \delta_i)] \in \mathbb{G}^4$.
5. Generate $(n+1)$ random values $r_i \in \mathbb{Z}_q$ such that $\prod_{i=1}^{n+1} r_i = 1$.
6. Let $c'_i = \mathsf{uRe}([(r_i\alpha_i, \beta_i); (\gamma_i, \delta_i)])$ for each $i \in [1, n]$.
7. Let $c'_{n+1} = \mathsf{uE}_y(r_{n+1})$.
8. Return a random permutation of $(c'_1, \ldots, c'_{n+1})$.

Figure 3-7: Commutative encryption based on the GJJS cryptosystem

each $r_j$ into each corresponding $c_j$'s $\alpha_j$ value, then universally re-randomize each modified ciphertext. Recall that a ciphertext will have the property that $\prod_{i=1}^{n} \alpha_i/\beta_i^{x_i} = m$. Thus, by the following, it is the case that correctness of decryption will be preserved:

$$m = \prod_{i=1}^{n} \frac{\alpha_i}{\beta_i^{x_i}} = \frac{\alpha_k}{\beta_k^{x}} \prod_{j \neq k}^{n} \frac{\alpha_j}{\beta_j^{x_j}} = \prod_{j \neq k}^{n} r_j \frac{\alpha_j}{\beta_j^{x_j}}$$

**Theorem 10** *$\mathcal{C}_{com}$ is efficiently cascadable.*

**Proof:** A $\mathsf{E}$ input in $C_i$ will be a permuted list of $i$ $\mathsf{uE}$ ciphertexts, each of size $\mathrm{poly}(\kappa)$. The output will simply be a list of $(i+1)$ $\mathsf{uE}$ ciphertexts, so the difference in size will be $\mathrm{poly}(\kappa)$. □

### 3.5.3 $\mathcal{C}_{com}$ exhibits rewrite fidelity

**Theorem 11** *$\mathcal{C}_{com}$ exhibits rewrite fidelity.*

**Proof of Theorem 11:** Rewrite fidelity is a required property for cascadable cryptosystems and will be important in proving the cascadable semantic security of $\mathcal{C}_{com}$. We show that for any string $s \in \Gamma^*$ and any message $m \in M$, evaluating $s(m)$ yields the same distribution as $\bar{s}(m)$. We first characterize the distribution $\bar{s}(m)$, then will inductively show that $\bar{s}(m) = s(m)$.

**Definition 12 (Effective Key Multi-set)** *For a string $s \in \Gamma^*$, if $\nu(s)$ is finite then the effective key multi-set $K(s) = \{k | \mathsf{e}_k \in \bar{s}\}$. If $\nu(s) = \infty$, then $K(s) = \{\perp\}$.*

COMMUTATIVE DECRYPTION $\mathrm{D}(x, z)$:

1. Input is a secret key $x \in \mathbb{Z}_q$ and a ciphertext $z \in \mathbb{G}^{4n}$, for $n \geq 1$.
2. Otherwise, return $\perp$.
3. Parse $z$ as a list of $n$ elements of the form $c_i = [(\alpha_i, \beta_i); (\gamma_i, \delta_i)] \in \mathbb{G}^4$.
4. Search for an element $c_k$ such that $\gamma_k / \delta_k^x = 1$.
5. If no such element exists, return $\perp$.
6. If $n = 1$, return $m = \alpha_k / \beta_k^x$.
7. Otherwise, choose $(n-1)$ random elements $r_j$ such that $\prod_{j \neq k}^{n} r_j = \alpha_k / \beta_k^x$.
8. Let $c'_j = \mathsf{uRe}([(r_j \alpha_j, \beta_j); (\gamma_j, \delta_j)])$ for $j \in [1, k-1] \cup [k+1, n]$.
9. Return a random permutation of $(c'_1, \ldots, c'_{k-1}, c'_{k+1}, c'_n)$.

Figure 3-8: Commutative decryption based on the GJJS cryptosystem

**Definition 13 ($T(K, m)$ Distribution)** *Define $T(K, m)$ to be the uniform distribution over all permutations of sets of ciphertexts $\{\mathsf{uE}_k(r_i) \mid k \in K\}$ for every $(r_1, \ldots, r_{|K|}) \in \mathbb{G}^{|K|}$ such that $\prod_{i=1}^{|K|} r_i = m$. If $K = \emptyset$ then $T(K, m) = \{m\}$ and if $K = \{\perp\}$, then $T(K, m) = \{\perp\}$.*

The following theorem, Theorem 12, is necessary in the proof of Theorem 11.

**Theorem 12** *In $\mathcal{C}_{com}$, $\forall s \in \Gamma^*$, $\forall m \in M$, $\bar{s}(m) = T(K(s), m)$.*

**Proof of Theorem 12:** Clearly, Theorem 12 holds for the base case where $\bar{s} = \epsilon$. Note that in $\mathcal{S}_{com}$, if $\bar{s}$ contains a $\mathsf{d}$ symbol, then $\nu(\bar{s}) = \infty$ since there does not exist any string $t$ such that $\overline{ts} = \epsilon$. Otherwise, $\nu(\bar{s})$ is finite only if $\bar{s}$ contains only $\mathsf{e}$ symbols. This proof will look at each of these cases and show that $\bar{s} = T(K(s), m)$.

**Case 1 ($\exists \mathsf{d} \in \bar{s}$):** Since no rules in $\mathcal{S}_{com}$ may be applied to $\bar{s}$, it cannot contain any substrings of the form $\mathsf{d}_i\mathsf{e}_i$ or $\mathsf{d}_i\mathsf{e}_j$. So any $\mathsf{d}$ symbols must be the right-most symbols in $\bar{s}$. Therefore, the first symbol in such a $\bar{s}$ that is evaluated on any message $m$ must be a $\mathsf{d}$. By the definition of $\mathsf{E}$, $\mathsf{d}_k(m) = \perp$ for all messages and keys. Thus, if $\bar{s}$ contains a $\mathsf{d}$ symbol, then $\bar{s}(m) = \{\perp\}$. Because $\nu(\bar{s}) = \infty$, it is the case that $\bar{s}(m) = T(K(s), m) = \{\perp\}$.

**Case 2 ($\neg\exists \mathsf{d} \in \bar{s}$):** In this case, $\bar{s}$ consists entirely of $\mathsf{e}_k$ symbols. Evaluating a string of $\mathsf{e}_k$ symbols will encrypt random values whose product is $m$ under each key in $K(s)$, and permute the output. This is exactly the distribution described by $T(K(s), m)$. (End proof of Theorem 12.) $\square$

We will now make an inductive hypothesis that $\forall s$ such that $|s| \leq n$, that $\forall m \in M$, $s(m) = \bar{s}(m)$. This hypothesis will show that for all strings $t$ of length $n+1$, $t(m) = \bar{t}(m)$. Consider the base case of $n = 0$, or $s = \epsilon = \bar{s}$. Clearly $s(m) = \{m\} = \bar{s}(m)$. The hypothesis holds for $n = 1$, where either $s = \mathsf{e}_k$ or $s = \mathsf{d}_k$, for some key $k$. Evaluating $\mathsf{e}_k(m)$ means encrypting the message $m$ under the key $k$, which yields an element from $T(\{k\}, m) = \bar{s}(m)$. Evaluating $\mathsf{d}_k(m)$ would yield $\perp$, which is also identical to $\bar{s}(m)$.

Assume the hypothesis holds for some arbitrary $n$, i.e. $\forall s \in \Gamma^n$, $\forall m \in M$ assume that $s(m) = \bar{s}(m) = T(K(s), m)$. Now consider a string $t = \mathsf{e}_k s$ or $t = \mathsf{d}_k s$. First, note that if $\nu(s) = \infty$, then $s(m) = \{\perp\}$ and clearly $t(m) = \{\perp\}$. Since $\nu(t) = \infty$, $\bar{t}(m) = \{\perp\}$. Thus, induction holds if $\nu(s) = \infty$.

If $\nu(s)$ is finite then $K(s)$ is a multi-set of keys. Under $\mathcal{S}_{com}$ if $t = \mathsf{e}_k s$, then $\bar{t} = \mathsf{e}_k \bar{s}$. This is because no decryption operation removes $\mathsf{e}_k$ and there are no rewrite rules that replace a substring of the form $\mathsf{ed}$ or $\mathsf{ee}$. By the definition of $\mathsf{E}$, applying $\mathsf{e}_k$ to an element in $s(m) = T(K(s), m)$ will output an element in $T(K(s) \cup \{k\}, m) = T(K(t), m) = \bar{t}(m)$. Thus, the induction holds if $\nu(s)$ is finite and $t = \mathsf{e}_k s$.

The remaining case is if $\nu(s)$ is finite and $t = \mathsf{d}_k s$. This is a trickier case because in $\mathcal{C}_{com}$, attempting to decrypt an element $c \in s(m)$ with a key $k$ that it has not been encrypted with will output a $\perp$ symbol.

Consider first the case where $\mathsf{e}_k \in \bar{s}$. Then an element $c \in \bar{s}(m)$ will contain a ciphertext encrypted under $k$. $\mathsf{D}$ will search $c$ for that ciphertext, decrypt it, randomly split its contents among the remaining ciphertexts, and output the remaining ciphertexts. Thus, it is the case that $t(m) = T(K(s) - \{k\}, m)$.

Since $\nu(s)$ is finite, $\bar{s}$ will contain strictly $\mathsf{e}$ symbols. The canonical form of $\mathsf{d}_k \bar{s}$ will be obtained by applying $\mathsf{d}$-skip rules $\mathsf{d}_k \mathsf{e}_j \rightarrow \mathsf{e}_j \mathsf{d}_k$ until a decryption rule $\mathsf{d}_k \mathsf{e}_k$ can be applied. Thus, $\bar{t}$ will be identical to $\bar{s}$, except with the first occurring $\mathsf{e}_k$ symbol spliced out. Thus, $K(t) = K(s) - \{k\}$. Since $\bar{t}(m) = T(K(t), m)$, it is the case that $t(m) = \bar{t}(m)$.

The final case is if $t = \mathsf{d}_k s$ and $\mathsf{e}_k \notin \bar{s}$. While evaluating $\mathsf{d}_k$ on some $c \in \bar{s}(m)$, since $k \notin K(s)$ the function $\mathsf{D}$ will not locate a decryptable ciphertext block and will output $\perp$, so $t(m) = \{\perp\}$.

Now one must consider the canonical form of $t = \mathsf{d}_k s$. Clearly, it is the case that $t \xrightarrow{*} \mathsf{d}_k \bar{s}$. Any rules applied at this point must involve $\mathsf{d}_k$, since $\bar{s}$ is canonical. Because $\nu(s)$ is finite, $\bar{s}$ contains strictly $\mathsf{e}$ symbols and because it was assumed $\mathsf{e}_k \notin \bar{s}$, only $\mathsf{d}$-skip rules can be applied. These will be applied until they no longer can, so $\bar{t} = \bar{s}\mathsf{d}_k$ and $\nu(t) = \infty$. By definition, $K(t) = \{\perp\}$ and $\bar{t}(m) = \{\perp\}$. Thus $t(m) = \bar{t}(m)$.

For both finite and infinite $\nu(s)$ for $n$-length $s$, it was shown that for both $t = \mathsf{e}_k s$ and $t = \mathsf{d}_k s$ that $t(m) = \bar{t}(m)$ for all $m \in M$. Thus, for any $(n+1)$-length string, the inductive hypothesis holds. Since $n$ was chosen arbitrarily, it is the case that $\forall s \in \Gamma^*$ $\forall m \in M$, $s(m) = \bar{s}(m)$, that is, rewrite fidelity holds. (End proof of Theorem 11.) $\square$

### 3.5.4 $\mathcal{C}_{com}$ is cascadable semantically secure

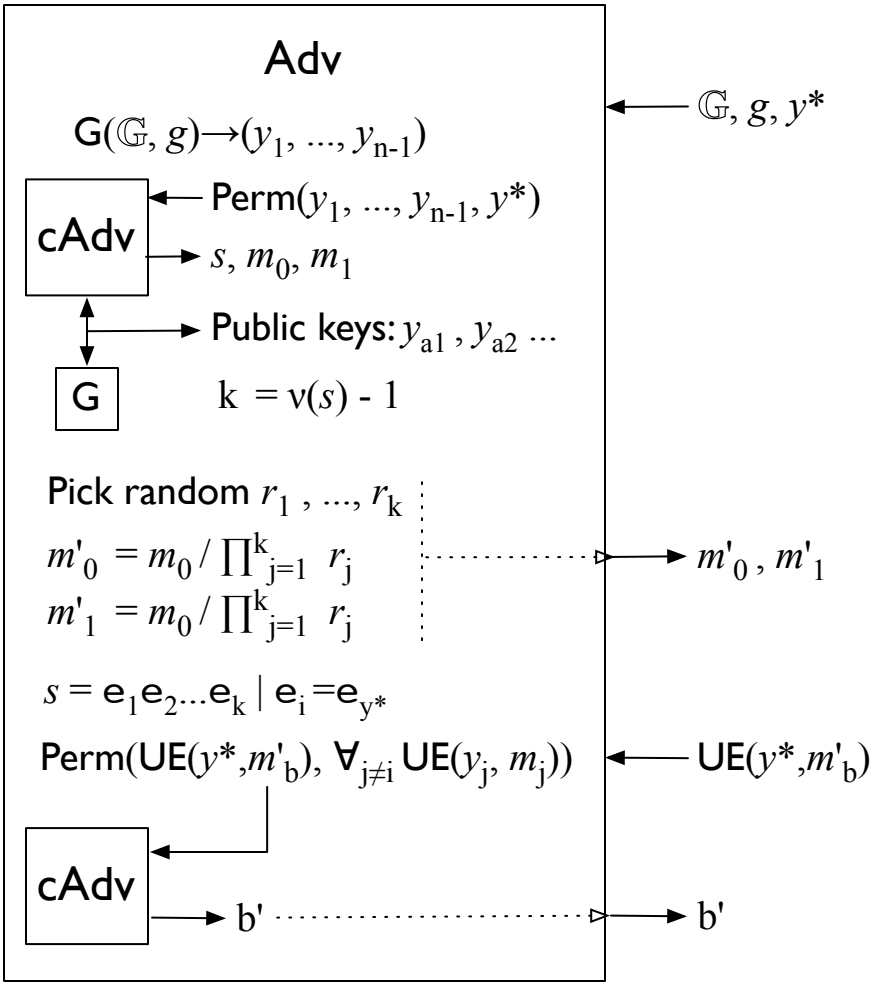


Figure 3-9: Reduction of IND-CPA security to CIND-CPA security in $\mathcal{C}_{com}$

**Theorem 13** *$\mathcal{C}_{com}$ is cascadable semantically secure.*

**Proof:** This proof will reduce a traditional IND-CPA experiment in the underlying GJJS cryptosystem $(\mathsf{SysGen}, \mathsf{G}, \mathsf{uE}, \mathsf{uD})$ to a CIND-CPA experiment with $\mathcal{C}_{com}$. The reduction described in this proof is illustrated in figure 3-9. Assume that GJJS is semantically secure under standard complexity assumptions. Assume, for the sake of contradiction, that some probabilistic polynomial time adversary $c\mathsf{Adv}$ succeeds in the cascadable CIND-CPA experiment $\text{Exp}^{CIND-CPA}_{\mathcal{C}_{com}, c\mathsf{Adv}}[\kappa, n]$ with non-negligible, polynomial advantage $1/\text{poly}(\kappa)$ for some $n = \text{poly}(\kappa)$.

With cryptosystem $\mathcal{C}_{com}$, the cascadable CIND-CPA experiment will first input the system parameters $(\mathbb{G}, g)$ and a list of public keys $(y_1 = g^{x_1}, \ldots, y_n = g^{x_n})$ to $c\mathsf{Adv}$. Then $c\mathsf{Adv}$ will output $m_0, m_1 \in \mathbb{G}$ and some $s \in \Gamma^*$. Finally, on input $c \in s(m_b)$, $c\mathsf{Adv}$ will correctly identify $b$ with probability at least $1/2 + 1/\text{poly}(\kappa)$.

We now define an adversary $\mathsf{Adv}$ that will use $c\mathsf{Adv}$ to obtain an advantage in its own traditional IND-CPA experiment. $\mathsf{Adv}$ will first receive a *single* public key $y^*$ as

input, then output two messages $m_0, m_1 \in \mathbb{G}$, obtain $c = \mathsf{uE}_{y^*}(m_b)$ and finally output a guess bit $b'$.

Adv will call to G to generate $(n-1)$ other public keys, then will randomly permute all $n$ keys before providing them to $c$Adv as input. These $n$ keys come from an identical distribution as $c$Adv would typically expect, so will not affect its advantage.

$c$Adv will then output $s$, $m_0$ and $m_1$. The canonical form $\bar{s}$ must contain at least one unmatched e symbol corresponding to one of the $n$ keys that Adv just gave as input. Since these symbols were randomly generated and randomly permuted, there is at least a $1/n$ chance that $y^*$ is among the unmatched e symbols in $\bar{s}$. Assume that this is the case.

Let $k = \nu(s) - 1$. This is the net number of encryptions *besides* $y^*$ represented by $\bar{s}$. Adv will choose $k$ random values in $\mathbb{G}$ denoted $r_1, \ldots, r_k$ and will output messages $m'_0 = m_0 / \prod_j r_j$ and $m'_1 = m_1 / \prod_j r_j$ in its own IND-CPA experiment. Adv then receives $c^* = \mathsf{uE}_{y^*}(m'_b)$ in response.

Adv now encrypts each $r_j$ value under the respective keys $\{j | \mathsf{e}_j \in \bar{s}\}$, i.e. $c_j = \mathsf{uE}_j(r_j)$. Adv will then randomly permute the ciphertexts $(c_1, \ldots, c_k, c^*)$ and present it to $c$Adv as a $\mathcal{C}_{com}$ commutative ciphertext. This ciphertext is a randomly permuted list of uE ciphertexts encrypted under each public key represented by $\bar{s}$, such that the product of their decrypted plaintexts is $m_b = m'_b \prod_j r_j$. In other words, this ciphertext comes from the distribution $\bar{s}(m_b)$, which by rewrite fidelity, is identical to $s(m_b)$.

Adv may then submit this ciphertext to $c$Adv, which distinguishes $b$ with some non-negligible advantage $1/\text{poly}(\kappa)$ by assumption. Since this reduction is conditioned on $\mathsf{e}_{y^*}$ being an unmatched symbol in $\bar{s}$, Adv will maintain at least a $1/(n \cdot \text{poly}(\kappa))$ advantage in its own traditional IND-CPA experiment. Since $n$ is polynomial in $\kappa$, this would still constitute a polynomial advantage in distinguishing GJJS ciphertexts. That violates the assumption that GJJS is semantically secure. Therefore, no such probabilistic polynomial time $c$Adv can exist and $\mathcal{C}_{com}$ must be cascadable semantically secure. □

### 3.5.5 $\mathcal{C}_{com}$ historical revelation properties

Clearly, $\mathcal{C}_{com}$ is both height revealing and known secret key set revealing. Given a ciphertext $c \in \mathbb{G}^{4i}$, anyone can can determine $i$ with no $x$ advice. Similarly, given a set of secret keys $X$, an adversary may scan through a permuted list of uE ciphertexts searching for any $(\gamma, \delta)$ values such that $\gamma/\delta^x = 1$ for some $x \in X$. If any such $x$ values are found found, the adversary knows with high probability that some $y = g^x$ was used to encrypt the message, i.e. that $\mathsf{e}_y \in \bar{s}$.

However, $\mathcal{C}_{com}$ is *not* public key set revealing. This is due to the use of GJJS universally re-randomizable scheme. A semantically secure construction could simply use ElGamal encryption as the underlying scheme, and include the public key with each ciphertext, i.e. using $\mathsf{E}(y, m, r) = (y, g^r, my^r)$ as the underlying building block. This method was described in Section 3.4.4. However, that would clearly reveal which public keys have encrypted a particular ciphertext. Under standard complexity as-

sumptions, GJJS will hide which public keys have encrypted each ciphertext and $\mathcal{C}_{com}$ is not public key set revealing.

Finally, because uE ciphertext blocks are permuted on each operation, any information about the sequence of operations is lost. Encryptions under a set of keys yields the same distribution of ciphertexts, regardless of order of operations. A consequence is that $\mathcal{C}_{com}$ is neither last encryption revealing nor sequence revealing. Table 3.2 compares $\mathcal{C}_{com}$'s historical properties to $\mathcal{C}_{casc}$ and $\mathcal{C}_{xor}$.

| **Cryposystem Property** | $\mathcal{C}_{casc}$ | $\mathcal{C}_{com}$ | $\mathcal{C}_{xor}$ |
|---|---|---|---|
| Cascadable Semantically Secure | Yes | Yes | No |
| Efficiently Cascadable | Maybe | Yes | Yes |
| Height Revealing | Maybe | Yes | No |
| Last Encryption Revealing | Maybe | No | No |
| Known Secret Key Set Revealing | Last key only | Yes | No |
| Known Public Key Set Revealing | No | No | N/A |
| Sequence Revealing | No | No | No |

Table 3.2: Various properties of $\mathcal{C}_{casc}$, $\mathcal{C}_{com}$, and $\mathcal{C}_{xor}$

## 3.6 Conclusion and Open Questions

This chapter presented a new security framework for cascadable cryptography that is particularly useful for commutative cryptosystems. The new cascadable semantic security definition works sensibly in spite of any historical properties that might be revealed by a ciphertext. By contrast, traditional semantic security definitions may fail miserably in settings where the message space may contain history-revealing ciphertexts. To illustrate the practical use of this model, this work offers two practical constructions that are provably cascadable semantically secure. This work raises many interesting open questions:

- Can we define a notion of CCA security for cascadable, and specifically, commutative cryptosystems? How would this notion relate to Re-playable CCA (RCCA) security [21]?

- Can we fully characterize which string rewrite models could conceivably be part of a cascadable semantically secure cryptosystem? Section 3.3.2 briefly discusses this issue.

- Can we prove that if a cryptosystem is semantically secure over all message domains $C_s$, then it is cascadable semantically secure?

- How can we accommodate non-terminating rewrite system models? Section 3.4.3 proposes a class rewrite system approach.

- Homomorphism, blinding operations, threshold operations, signatures, re-randomization, or hashing might be modeled with term rewrite systems. How can we adapt the definitions in this chapter to more general term rewrite systems?
- Which historical revelation properties are interesting in either a practical or a theoretical sense?
- Besides commutativity, which sub-classes of cascadable cryptosystems are of particular theoretical or practical interest?
- Can we implement a cascadable semantically secure cryptosystem with $\mathcal{S}_{casc}$ plus rules of the form $\mathsf{e}_k\mathsf{d}_k \rightarrow \epsilon$ as the rewrite model? This type of cryptosystem was proposed by Diffie and Hellman [35].
- Are there any constructions of cascadable semantically secure cryptosystems based on $\mathcal{S}_{casc}$ that are not height revealing?

# Chapter 4

# Honest-Verifier Private Disjointness Testing

This chapter presents an efficient construction of a *private disjointness testing* protocol that is secure against malicious provers and honest-but-curious (semi-honest) verifiers, without the use of random oracles. In a completely semi-honest setting, this construction implements a *private intersection cardinality* protocol. This work appears in [59] and was developed with Susan Hohenberger, so will be referred to as HW.

Suppose two parties, Alice and Bob, each have a private database of values, respectively denoted $A$ and $B$, where the set cardinalities $|A|$ and $|B|$ are publicly known. Alice wishes to learn whether their two sets are disjoint, that is, whether $A \cap B = \emptyset$, or how large the intersection is, that is, $|A \cap B|$. In doing so, Alice cannot reveal information about her set $A$ to Bob, who in turn does not want to reveal information about his set $B$, other than the bit $A \cap B \stackrel{?}{=} \emptyset$ or, perhaps, the size of the intersection $|A \cap B|$. These are respectively the *private disjointness testing* (PDT) and *private intersection cardinality* (PIC) problems.

For example, Alice may be a law enforcement agent ensuring that no suspects under investigation purchased tickets on a flight operated by Bob. Alice cannot simply reveal her list of suspects to Bob without compromising her investigation. Nor can Bob disclose any passenger names without explicit subpoenas. Yet, both parties have an interest in alerting Alice whether any suspects are on Bob's flight.

As another example, suppose Bob wants to anonymously login to Alice's system. Bob needs to prove that one of his identities in a set $B$, which may be a singleton, is among the set of Alice's valid users, denoted $A$. Alice should be convinced that Bob is a legitimate user, without learning which specific user he is. Thus, both parties wish to determine whether $|A \cap B| \neq 0$.

These types of private set operations may be implemented by several existing techniques. Private set operations may be viewed as a general two-party secure computation problem, solvable by classic secure multiparty computation techniques [51, 125]. Zero-knowledge sets due to Micali, Rabin and Killian [85], support private operations like disjointness testing, set union, and set intersection.

Unfortunately, these techniques have remained unused in practice due to their

high computation, communication, and implementation costs. Oblivious polynomial evaluation protocols, such as those due to Naor and Pinkas [90], may also be applied to private set operations. However, using generalized oblivious polynomial evaluation for private set operations is inefficient in comparison to specialized protocols.

This chapter builds on specialized private set operation protocols recently developed by Freedman, Nissim, and Pinkas (FNP) [46], and Kiayias and Mitrofanova (KM) [71], and offers a new construction that is more efficient in a malicious-prover setting. When both parties are honest-but-curious (semi-honest), the Hohenberger and Weis (HW) construction is a *private intersection cardinality* protocol, where a verifier party (who is played by Alice in the above examples) learns $|A \cap B|$. The efficiency in this setting is equivalent to both FNP and KM, but is based on a different complexity assumption.

Note that in the context of "honest-verifier", we are using the term "honest" interchangeably with "semi-honest". This means the verifier is honest-but-curious about the values it receives and while abiding by the protocol, may examine any received values further to try to learn more about $B$. The notion of semi-honest or honest-but-curious was introduced in [51]

The HW construction improves on existing results in settings where the prover is malicious and the verifier is honest-but-curious. In this malicious-prover setting, the HW construction implements a *private disjointness testing* protocol. A malicious, polynomial-time bounded prover able to send arbitrary messages cannot convince a verifier that their sets intersect, unless they actually do. In the anonymous login example, Bob will not be able to login unless he possesses a legitimate identity string.

The HW honest-but-curious (semi-honest) private intersection cardinality protocol presented in this chapter *as is* becomes a private disjointness testing protocol in the malicious-prover setting. By contrast, previous works require additional computations, such as adding zero-knowledge proofs [71] or homomorphic encryptions [46], to be made secure in a malicious-prover setting. Moreover, both FNP and KM require random oracles to be made secure in the presence of a malicious prover, whereas the HW construction does not.

### 4.0.1 The FNP Protocol Paradigm

The FNP protocol [46] is quite intuitive and simple, and is the design paradigm used in both the KM and HW protocols. An FNP invocation where Bob has a singleton set is informally outlined in figure 4-1. To provide further technical details, suppose $(G, E, D)$ is a semantically-secure homomorphic encryption scheme. Let $\mathcal{V}$ have set $A = \{a_1, \ldots, a_n\}$ and $\mathcal{P}$ have set $B = \{b_1, \ldots, b_m\}$.

As shown in figure 4-1, the verifier (also known as Alice) first selects a random constant or irreducible polynomial $G(x)$ (i.e. $G(x)$ will have no roots). The verifier than computes $f(x) = G(x) \cdot (\prod_{a_i \in A}(x - a_i)) = \sum \alpha_i x^i$. Note that the roots of $f$ are exactly the values in the set $A$. The verifier then encrypts the $\alpha$ coefficients of $f$ under a public key $pk$ that she chooses, and sends them to the prover. That is, $\mathcal{V}$ encrypts each coefficient as $c_i = E_{pk}(\alpha_i)$ with a homomorphic cryptosystem such as Paillier's [94, 95].

THE FNP PROTOCOL:

1. $\mathcal{V}$ chooses a random constant or irreducible polynomial $G(x)$.
2. $\mathcal{V}$ computes $f(x) = G(x) \cdot (\prod_{a_i \in A}(x - a_i)) = \sum \alpha_i x^i$.
3. If any $\alpha_i = 0$, restart the protocol.
4. $\mathcal{V}$ encrypts the coefficients of $f(x)$ with a homomorphic encryption scheme and sends the encryptions $c_i = E(\alpha_i)$ to $\mathcal{P}$.
5. Using the homomorphic properties of $E$, $\mathcal{P}$ obliviously evaluates $f(x)$ at some value $b$, obtaining $E(f(b))$.
6. $\mathcal{P}$ randomizes his evaluation as $c = E(Rf(b))$ and sends it to $\mathcal{V}$.
7. $\mathcal{V}$ decrypts $c$. If $D(c) = 0$, $\mathcal{V}$ knows $\mathcal{P}$'s value intersects with $A$.

Figure 4-1: An overview of the Freedman, Nissim, and Pinkas (FNP) protocol

Recall that homomorphic cryptosystems like Paillier's allow a party given $E_{pk}(x)$ and $E_{pk}(y)$ to obliviously compute $E_{pk}(x) \cdot E_{pk}(y) = E_{pk}(x+y)$, or to compute $E_{pk}(x)^z = E_{pk}(x \cdot z)$, where $z$ is some constant. Note that given the encryptions $c_i$, these homomorphic operations are sufficient to obliviously evaluate the polynomial $f$. For example, the encryptions $c_0 = E_{pk}(4)$ and $c_1 = E_{pk}(3)$ commit the polynomial $f(x) = 3x + 4$. A second party may evaluate this at a particular value $x = 2$, by computing

$$c_1^2 \cdot c_0 = E_{pk}(3 \cdot 2) \cdot E_{pk}(4) = E_{pk}(6 + 4) = E_{pk}(10) = E_{pk}(f(2))$$

Thus, given coefficients encrypted as $c_i$ values, the prover (Bob) may obliviously evaluate $f(b_i)$ for each element $b_i \in B$. Note that if $b_i \in A$, then $f(b_i) = 0$. The prover will now randomize all his obliviously evaluated $f(b_i)$ values by homomorphically multiplying them by a random nonzero value. That is, he computes $E_{pk}(f(b_i))^r = E_{pk}(r \cdot f(b_i))$ where $r$ is a random nonzero value. Thus, if $f(b_i) = 0$, then the encryption of $E_{pk}(r \cdot f(b_i)) = E_{pk}(0)$. Otherwise, $E_{pk}(r \cdot f(b_i))$ is some random value. This hides any information about elements in the prover's set that are *not* in the verifier's set.

The prover now sends each of these encrypted oblivious evaluations $E(r_i \cdot f(b_i))$ to the verifier. The verifier then decrypts and tests whether any of the resulting plaintexts are zero. If $b_i \in A$, then $f(b_i) = 0$, so if any decrypted values are zero, then the verifier believes there is an intersection with the prover's set. Note that the original FNP protocol reveals the elements in the intersection of the two sets, by having the prover return the ciphertext $E_{pk}(r \cdot f(b_i) + b_i)$ instead. Thus if $f(b_i) = 0$, the verifier will get the actual elements of the intersection – not just the cardinality.

We focus on applications where the prover explicitly does not want to reveal anything about his set, except the size or existence of the intersection. For instance,

the anonymous login application cannot have the verifier learn the actual intersection values. This chapter will only focus on the private intersection cardinality protocol version of FNP, although finding actual intersection values will be discussed further in Section 4.6.6.

In the KM protocol [71], the same techniques as FNP are applied, except that it uses a new primitive called *superposed encryption* based on Pedersen commitments [96]. Superposed encryption is closely related to a homomorphic ElGamal variant first used in voting schemes by Cramer, Gennaro, and Schoenmakers [29]. In the KM protocol the prover returns to the verifier a single ciphertext $E_{pk}(r \cdot |A \cap B|)$, where $r$ is a random value. Thus, this is specifically a PDT protocol rather than a PIC protocol. The verifier accepts if the ciphertext decrypts to zero and rejects otherwise.

Both the FNP and KM constructions, based on Paillier's homomorphic encryption [94, 95] and Pedersen's commitment scheme [96], suffer from a crucial flaw: *malicious adversaries may simply encrypt or commit to zero values.* For instance, in the FNP case, someone can simply encrypt 0 with the public key and convince the verifier that an intersection exists when it does not.

This is a critical failure which both FNP and KM immediately recognize and address. To cope with malicious provers, FNP proposes a fix that relies on the random oracle model (ROM), despite its inherent problems [5, 20].

The FNP fix essentially binds the randomness used to prepare encryptions of coefficients to the values of the coefficients themselves and is quite simple. Assuming random oracles $G$ and $H$, the prover will first generate a random $s$ and compute $(r, r', r'') \leftarrow G(s)$. Then the prover will oblivious compute $c = E_{pk}(r, r' \cdot f(x) + s)$ as well as a check value $h = H(r'', x)$.

Upon decrypting $c$, if $f(x) = 0$, the verifier will obtain $s$ and be able to search for a $x \in A$ that is consistent with $s$ and $h$. Since the ciphertext $c$ depends entirely on $s$ and $x$, the verifier should be able to re-create the prover's computation and produce the identical ciphertext $c$. Note that this returns the *intersection* to an honest verifier.

Otherwise, if $f(x) \neq 0$, the randomizing factor $r'$ will hide any information about $f(x)$ and $s$. This construction prevents a malicious prover from simply encrypting a 0 value directly. This is similar in many respects to constructions of adaptive chosen-ciphertext (CCA) secure cryptosystems in the random oracle model [6, 47].

Fixing KM against malicious adversaries requires both the use of random oracles as well as universally-composable (UC) commitments [19], which require the assumption of a common reference string. While relatively efficient, the best known UC commitment schemes are interactive and would increase communication complexity by a quadratic factor [18, 22, 31].

The weakness of FNP and KM in the face of malicious provers begs the question: Can we implement an efficient private disjointness testing protocol without the use of random oracles or costly sub-protocols? This chapter answers this question in the affirmative.

### 4.0.2 Overview of the HW Construction

This section provides intuition for understanding the Hohenberger and Weis (HW) construction in the context of prior work. Essentially, the main difference is that in both the FNP and KM protocols, a prover convinces a verifier to accept by returning an encryption of zero. If the prover was honest, then if the verifier receives an encryption of a zero value, it implies some element in $\mathcal{P}$'s set is also in $\mathcal{V}$'s set. However, if the prover is malicious, then he can easily encrypt a zero value from scratch and send it to the verifier. To prevent this, both FNP and KM must add costly computations to check that the prover follows a specified protocol.

To cope with malicious provers, the HW construction essentially substitutes a cryptographic primitive dubbed "testable and homomorphic commitment" in the place of Paillier's homomorphic encryption. Instead of encryptions of zero, elements belonging to the intersection of the two sets will be encoded to have a specific order in a multiplicative group. In other words, a prover convinces a verifier that an intersection exists by returning elements of a specific order.

The necessary complexity-theoretic assumptions are that it is hard to for a prover to decide whether group elements belong to a particular subgroup of unknown order, and that it is hard to compute elements in the subgroup. Under this *subgroup computation assumption*, computing an element of this order is hard for a prover, unless he correctly follows the protocol (and there *is* a non-empty intersection). Thus, in the malicious-prover setting, the HW construction is sound by default, whereas FNP and KM must augment their protocols with costly computations in the random oracle model.

In the HW construction presented in Section 4.3, the verifier begins, as in FNP, by selecting a random polynomial $f(\cdot)$ whose roots correspond to set $A$. The verifier computes a *testable and homomorphic commitment* (THC) of each coefficient, which is essentially a BGN encryption [13] set in group $\mathbb{G}$, which has order $n = pq$ where $p$ and $q$ are large primes.

For each element $b_i \in B$, the prover uses THCs to compute a value that will be a random element in $\mathbb{G}$ if $b_i \notin A$ or will be a random element of order $p$ if $b_i \in A$. The verifier, with knowledge of $p$ and $q$, can test the order of each element returned by the prover. In this way, the verifier learns the cardinality of the intersection, just as in FNP.

The main benefit, however, is that a malicious prover cannot, under the subgroup computation problem, compute an element of order $p$ from scratch. As will be proven in Section 4.4.1, the HW construction remains sound in the malicious-prover setting without any augmentation. As in the FNP PDT protocol, the verifier can *potentially* learn the cardinality of the intersection, but is not *guaranteed* to do so when talking with a malicious prover. That is, if the prover happens to be honest, the verifier will learn the cardinality – but there is no way to know whether a prover is honest. Table 4.1 compares the behavior of FNP, KM, and the HW construction in different security settings.

### 4.0.3 Related Work

| Security Setting | FNP | KM | HW |
|---|---|---|---|
| **Semi-Honest** | Cardinality | Disjointness | **Cardinality** |
| **Malicious Prover** (Requirements) | Cardinality (ROM) | Disjointness (NIZK Proofs) (ROM) | **Disjointness** (**None**) |
| **Malicious Verifier** (Requirements) | Cardinality (Multiple Invocations) | Disjointness (UC-Commitments) (ROM) | See Section 4.6.1 |

Table 4.1: Three private set protocols compared in different security settings. ROM stands for "Random Oracle Model", NIZK for "Non-Interactive Zero Knowledge", and UC for "Universally Composable".

Kissner and Song [72] offer FNP-inspired schemes for solving several closely related privacy-preserving set operations like set disjointness, cardinality, and set union. They offer improved efficiency compared to FNP in the multiparty, honest-but-curious setting. Again, when translated to the malicious adversary model, their constructions require relatively costly zero-knowledge proof of knowledge sub-protocols. In all fairness, Kissner and Song address a richer set of problems than simple disjointness testing like set union, set intersection, and multiplicity testing. They also work in a multiparty model, so it is not surprising that their solutions require more computation.

Constructions from both Pedersen's commitment scheme [96] and Paillier's homomorphic cryptosystem [94, 95] are both closely related to the "testable and homomorphic commitment" primitive presented in Section 4.3.2.

The Subgroup Decision Assumption (SDA) and the Subgroup Computation Assumption (SCA) described in Section 4.1.2 are both crucial to proving security of the construction presented in this chapter. Yamamura and Saito apply the SDA to the private information retrieval problem [124]. The composite residuosity assumptions made by Paillier are also closely related.

A similar *bilinear* subgroup complexity assumption is made by Boneh, Goh, and Nissim for their 2DNF ciphertext evaluation scheme [13]. Groth, Ostrovsky, and Sahai also make the same complexity assumption to implement non-interactive zero knowledge proofs [55].

# 4.1 Preliminaries

## 4.1.1 Notation

Let $\mathbb{Z}$ be the integers. Let $\text{negl}(\cdot)$ be a *negligible* function such that for all polynomials $p(\cdot)$ and all sufficiently large $k \in \mathbb{Z}$, we have $\text{negl}(k) < 1/p(k)$. We will denote that two distributions $C$ and $D$ are perfectly indistinguishable using $C \approx D$ and computationally indistinguishable using $C \overset{c}{\approx} D$ notation. A $\mathcal{M}_{ppt}$ subscript will indi-

cate that a interactive Turing machine $\mathcal{M}$ runs in probabilistic polynomial time. The value $ord(x)$ will be the order of an element $x$. The transcript $\mathsf{View}^{\mathcal{M}}[\mathcal{M}(x)\mathcal{N}(y)]$ will represent the view of algorithm $\mathcal{M}$ after interacting with algorithm $\mathcal{N}$ on inputs $x$ and $y$, respectively. $\mathcal{M}$'s view includes its input, its randomness, and the public transcript of the protocol. We will denote a distribution of views over random inputs as $\{\mathsf{View}^{\mathcal{M}}[\mathcal{M}(x)\mathcal{N}(y)]\}$.

### 4.1.2 Complexity Assumptions

The complexity assumptions applied in the HW construction exist in various forms throughout the literature. The formalization here is closest to that of Yamamura and Saito [124]. Recently, Boneh, Goh, and Nissim introduced a stronger version of these assumptions for *bilinear* groups [13].

**Definition 14 (Subgroup Decision Assumption (SDA) [13, 124])** *Let $S(1^k)$ be an algorithm that produces $(\mathbb{G}, p, q)$ where $\mathbb{G}$ is a group of composite order $n = pq$, and $p < q$ are $k$-bit primes. Then, we say that the* subgroup decision *problem is hard in $\mathbb{G}$ if for all probabilistic polynomial time adversaries $\mathcal{A}$,*

$$\Pr\left[(\mathbb{G}, p, q) \leftarrow S(1^k);\ n = pq;\ x_0 \leftarrow \mathbb{G};\ x_1 \leftarrow x_0^q;\ b \leftarrow \{0,1\};\ b' \leftarrow \mathcal{A}(\mathbb{G}, n, x_b) : b = b'\right] \leq \frac{1}{2} + \mathrm{negl}(k).$$

Basically, the SDA means that given the description of a group $\mathbb{G}$, in the form of a generator $g$, and its order $n = pq$, a probabilistic polynomial-time adversary cannot distinguish random elements of order $p$ in $\mathbb{G}$ from random elements in $\mathbb{G}$. Clearly, if factoring is easy, then the SDA fails to hold. Similarly, someone able to compute discrete logarithms given $(\mathbb{G}, n, x)$ can decide this problem by computing $gcd(\log_g x, n)$, for some generator $g$. It is not clear how the SDA relates to the Decisional Diffie-Hellman (DDH) assumption.

Additionally, the security of the HW scheme requires the following computational assumption:

**Definition 15 (Subgroup Computation Assumption (SCA))** *Let $S(1^k)$ be an algorithm that produces $(\mathbb{G}, p, q)$ where $\mathbb{G}$ is a group of composite order $n = pq$, and $p < q$ are $k$-bit primes. Then, we say that the* subgroup computation *problem is hard in $\mathbb{G}$ if for all probabilistic polynomial time adversaries $\mathcal{A}$,*

$$\Pr\left[(\mathbb{G}, p, q) \leftarrow S(1^k);\ n = pq;\ x \leftarrow \mathcal{A}(\mathbb{G}, n) : ord(x) = p\right] \leq \mathrm{negl}(k).$$

An example group where these assumptions may be applied is a subgroup $\mathbb{G}$ of order $n = pq$, consisting of the quadratic residues of $\mathbb{Z}_{p'}$, where $p' = 2pq+1$ and $p', p, q$ are all primes. Of course, the HW construction can also operate over the bilinear groups where Boneh et al. [13] assume the subgroup decision problem is hard. It is not clear that the SDA assumption implies SCA, or vice versa, although a relation between the two seems plausible. Further exploration of both assumptions could be valuable in other schemes as well.

## 4.2 Problem Definitions

This section formally defines private intersection cardinality (PIC) and private disjointness testing (PDT) protocols. Let $1^k$ be a security parameter in unary. Let $Q$ be the domain of values for this protocol such that $|Q| \in \Theta(2^k)$. Let the universe $U$ be the set of all poly($k$)-sized subsets of $Q$. For sets $A \in U$ and $B \in U$, define the *disjointness predicate* $D(A, B) = (A \cap B = \emptyset)$, that is, $D(A, B)$ will have value 1 if and only if $A$ and $B$ are disjoint.

Let a verifier $\mathcal{V}$ and a prover $\mathcal{P}$ be two probabilistic polynomial time interactive Turing machines. Each party takes an element of $U$ (i.e. a poly($k$)-size subset of $Q$) as input. The interaction of $\mathcal{P}$ and $\mathcal{V}$ yields a result to $\mathcal{V}$ *only*.

### 4.2.1 Private Disjointness Testing Definition

**Definition 16 (Honest-Verifier Private Disjointness Testing)** *Two probabilistic polynomial time interactive Turing machines $(\mathcal{P}, \mathcal{V})$ define an* Honest-Verifier Private Disjointness Testing *protocol if the following conditions hold:*

1. **Completeness:** *For honest parties, the protocol works and the verifier learns the disjointness predicate; that is,*

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = D(A, B)\right] \geq (1 - \text{negl}(k))$$

   *where the probability is taken over the randomness of $\mathcal{P}$ and $\mathcal{V}$.*

2. **Soundness:** *For a random set $A \in U$, the probability that the prover will convince the verifier to accept is negligible; that is,*

$$\forall \mathcal{P}^*_{ppt}, \Pr_{A \in U}\left[\mathcal{P}^*\mathcal{V}(A) \neq 0\right] \leq \text{negl}(k)$$

   *where probability is taken over the choice of $A \in U$ and the randomness of $\mathcal{P}^*$ and $\mathcal{V}$.*

3. **Malicious-Prover Zero Knowledge (MPZK):** *A malicious prover learns nothing about the verifier's set; that is,*

$$\exists \mathcal{S}_{ppt},\ \forall \mathcal{P}^*_{ppt},\ \forall A \in U,\ \{\mathsf{View}^{\mathcal{P}^*}\left[\mathcal{P}^*\mathcal{V}(A)\right]\} \stackrel{c}{\approx} \{\mathsf{View}^{\mathcal{P}^*}\left[\mathcal{P}^*\mathcal{S}(1^{|A|})\right]\}$$

4. **Honest-Verifier Perfect Zero Knowledge (HVPZK):** *An honest-but-curious verifier learns nothing about the prover's set beyond the size of the intersection; that is,*

$$\exists \mathcal{S}_{ppt},\ \forall A \in U,\ \forall B \in U,\ \{\mathsf{View}^{\mathcal{V}}\left[\mathcal{P}(B)\mathcal{V}(A)\right]\} \approx \{\mathcal{S}(A, 1^{|B|}, 1^{|A \cap B|})\}$$

Note that an honest-but-curious verifier is allowed to *potentially* learn $|A \cap B|$, but he is not *guaranteed* to learn that value. One might define a stronger definition where rather than being provided $1^{|A \cap B|}$, the simulator would only be provided $D(A, B)$.

### 4.2.2 Private Intersection Cardinality Definition

**Definition 17 (Honest-Verifier Private Intersection Cardinality)** *An* Honest-Verifier Private Intersection Cardinality *protocol has the same setup as in Definition 16, except for the following differences:*

1. **Completeness:** *For honest parties, the protocol works and the verifier learns the cardinality predicate; that is,*

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = |A \cap B|\right] \geq (1 - \text{negl}(k))$$

   *where probability is taken over the randomness of $\mathcal{P}$ and $\mathcal{V}$.*

2. **Cardinality Soundness:** *A malicious prover can not convince an honest verifier that the cardinality is larger than it really is; that is,*

$$\forall \mathcal{P}^*_{ppt},\ \forall B \in U,\ \Pr_{A \in U}\left[\mathcal{P}^*(B)\mathcal{V}(A) > |A \cap B|\right] \leq \text{negl}(k)$$

   *where probability is taken over the choice of $A \in U$ and the randomness of $\mathcal{P}$ and $\mathcal{V}$.*

### 4.2.3 Informal Explanation of the Definitions

Completeness means that a correct execution between two honest parties will return the correct value to $\mathcal{V}$ with negligible chance for error. In a PDT protocol, the correct value is the disjointness predicate $D(A, B)$ and in a PIC protocol it is the intersection cardinality $|A \cap B|$.

PDT soundness implies that on a random input set $A \in U$, $\mathcal{V}$ has a negligible chance of obtaining a non-zero result when interacting with any malicious probabilistic polynomial-time prover $\mathcal{P}^*$. That is, unless $\mathcal{P}^*$ actually knows a value in $\mathcal{V}$'s set, or is extremely lucky, then $\mathcal{V}$ will not be fooled into thinking otherwise.

Neither the FNP nor KM protocols are sound by this definition. In those schemes, a verifier will believe that there is an intersection if it receives the value zero encrypted under a public-key. A malicious prover could trivially violate the soundness property by encrypting a zero value itself.

PIC soundness is similar to the PDT soundness definition, except that for any set $B$, and random set $A$, the protocol has a negligible chance of returning a value greater than $|A \cap B|$ to a verifier $\mathcal{V}$ interacting with $\mathcal{P}^*(B)$. The idea is that this prevents a malicious prover from doing trivial attacks like duplicating elements in its set $B$ to inflate the cardinality returned to the verifier. Of course, a malicious prover can always run the protocol on some subset of $B$, which would with high probability under-report the cardinality. This is unavoidable and is why cardinality soundness is only concerned with over-reporting the cardinality. As it turns out, this property will be the reason why the HW construction in Section 4.3 is *not* an Honest-Verifier Private Intersection Cardinality protocol. Section 4.5 will discuss this further.

Since a verifier is allowed to potentially learn $|A \cap B|$ in both the PDT and PIC protocols, the zero knowledge definitions presented in this chapter are the same. This relaxation appears in FNP as well, but not KM.

The Malicious-Prover Zero Knowledge (MPZK) property means that no probabilistic polynomial-time potentially malicious prover $\mathcal{P}^*$ can learn anything about a set $A$ from an interaction with $\mathcal{V}$ that it could not simulate on its own. In other words, the verifier's set, for example a database of passwords, remains hidden from even malicious provers. Here the distributions are computationally indistinguishable. Any action that $\mathcal{V}$ takes as a result of a successful protocol invocation, such as allowing $\mathcal{P}^*$ to anonymously login, is considered outside the protocol definition.

Finally, the Honest-Verifier Perfect Zero Knowledge (HVPZK) property implies that a probabilistic polynomial-time semi-honest verifier $\mathcal{V}$ does not learn anything about $B$ beyond the size of the set intersection. There is a subtle point here in the PDT protocol: the verifier is only *guaranteed* to learn the bit $D(A, B)$, but we allow an honest-but-curious verifier to *potentially* learn the size of the intersection. The flexibility suits the applications mentioned in the introduction. In fact, in the semi-honest setting, the distribution an adversary can simulate on its own is perfectly indistinguishable from a real transcript distribution.

The definitions in this chapter do not explicitly consider auxiliary inputs in the zero-knowledge definitions in this chapter. To do so, one need simply quantify over all polynomial-size advice strings and provide this string to both the party in question and the simulator.

## 4.3 HW Private Disjointness Testing

This section presents the Hohenberger-Weis (HW) construction, originally presented in [59], which implements an efficient PDT protocol. Section 4.4.1 proves that the HW construction securely meets the requirements of Definition 16. Overall, HW is very similar to those of Freedman, Nissim, and Pinkas (FNP) [46] and Kiayias and Mitrofanova (KM) [71].

FNP and KM respectively rely on Paillier's homomorphic encryption system [94, 95] and a Pedersen commitment variant [96] as underlying primitives. This chapter offers a new *testable and homomorphic commitment* (THC) primitive that will be used in a FNP-style oblivious polynomial evaluation scheme. The THC construction presented is reminiscent of both Paillier's and Pedersen's schemes. It is very similar to the encryption scheme for small messages due to Boneh, Goh, and Nissim (BGN) [13], but is used for the full range of messages.

The advantage of the HW construction is that it offers a stronger security guarantee than the basic FNP and KM protocols, with equivalent computation and communication costs. Although variants of both FNP and KM can be modified to offer stronger security, they require either the use of random oracles or significantly more computation.

### 4.3.1 Verifier System Setup

VERIFIER SYSTEM SETUP:

1. Run $S(1^k)$ to obtain $(\mathbb{G}, p, q)$.
2. Choose two random generators $g$ and $u$ from $\mathbb{G}$.
3. Compute $n = pq$ and $h = u^q$.
4. Publish $(\mathbb{G}, n)$ and keep $(p, q, g, h)$ private.

Figure 4-2: HW verifier system setup

As illustrated in figure 4-2, the HW construction is initialized by a verifier that selects some group of order $n = pq$, where $p$ and $q$ are secret large primes. The verifier will also select two random generators $g$ and $u$, and will compute $h = u^q$. Note that $h$ is a random generator of the subgroup of order $p$.

The verifier only needs to publish $\mathbb{G}$ and $n$. The prover will not know $p$, $q$, $h$ or even $g$. Learning $h$, $p$, or $q$ would allow a malicious prover to spuriously convince the verifier that an intersection exists.

### 4.3.2 Testable and Homomorphic Commitments

The public order $n$ and private values $g$ and $h$ may be used for a *testable and homomorphic commitment* (THC) scheme. This primitive will be the basis of the HW construction. Informally, a THC scheme supports the following operations:

- Commit: $\mathsf{Com}(m, r)$ a message $m$ with randomness $r$,
- Addition: For all $m, r, m', r'$, $\mathsf{Com}(m, r) \cdot \mathsf{Com}(m', r') = \mathsf{Com}(m + m', r + r')$,
- Constant Multiplication: For all $m, r, c$, $\mathsf{Com}(m, r)^c = \mathsf{Com}(cm, cr)$
- Equality Test: $\mathsf{Test}(\mathsf{Com}(m, r), x)$, returns 1 if $m = x$.

Testable and homomorphic commitments should be computationally hiding:

**Definition 18 (Testable and Homomorphic Commitment Hiding Property)** *Let $n$ be an integer, and let $a_0, a_1, r$ be values in $\mathbb{Z}_n^*$. Then, we say that a testable and homomorphic commitment $\mathsf{Com}$ set in a group $\mathbb{G}$ of order $n$ is* computationally hiding *over the distribution of $r$ if*

$$\forall a_0, a_1 \in \mathbb{Z}_n^*, \{\mathbb{G}, n, a_0, a_1, \mathsf{Com}(a_0, r)\} \overset{c}{\approx} \{\mathbb{G}, n, a_0, a_1, \mathsf{Com}(a_1, r)\}$$

The encryption scheme for small messages due to BGN is very similar to the HW construction, except for two differences. First, we provide the adversary with even less information about the commitment; that is, the values $g$ and $h$ remain private. Secondly, BGN allow and support bilinear map operations, whereas we do not consider them. Similarly to their scheme, the HW testable and homomorphic commitment primitive operates as shown in figure 4-3.

TESTABLE AND HOMOMORPHIC COMMITMENTS OPERATIONS:

1. **Setup:** Let $S(1^k)$ be an algorithm that outputs $(\mathbb{G}, p, q)$ where $\mathbb{G}$ is a group of composite order $n = pq$, and $p < q$ are $k$-bit primes. Let $g, u$ be random generators of $\mathbb{G}$ and let $h = u^q$. Publish $n$; keep all else private.
2. **Commit:** Given $m$ and $r \in \mathbb{Z}_n^*$, compute: $\mathsf{Com}(m, r) = g^m h^r$
3. **Addition:** $\mathsf{Com}(m, r) \cdot \mathsf{Com}(m', r') = g^{m+m'} h^{r+r'} = \mathsf{Com}(m + m', r + r')$
4. **Constant Multiplication:** $\mathsf{Com}(m, r)^c = g^{cm} h^{cr} = \mathsf{Com}(cm, cr)$
5. **Equality Test:** If $\mathsf{Test}(\mathsf{Com}(m, r)) = (g^m h^r / g^x)^p = (g^p)^{m-x} = 1$, output 1; else, output 0.

Figure 4-3: Testable and homomorphic commitment construction

**Lemma 4** *The testable and homomorphic commitment scheme described in figure 4-3 is computationally hiding, i.e., it satisfies definition 18.*

This lemma follows, more or less, from the semantic security of the encryption scheme of Boneh, Goh, and Nissim. For completeness, however, Section 4.4.1 will prove that this construction is computationally hiding.

### 4.3.3 Oblivious Polynomial Evaluation

Suppose a party knowing $h$ has some polynomial $f(x) = \sum \alpha_i x^i \in \mathbb{Z}_q[x]$. This party can publish commitments to $f$'s coefficients as $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i}$, where $\gamma_i$ values are random. Let $s = \lceil \sqrt{n} \rceil$. Assuming $p$ and $q$ are not twin primes, we have that $p < s < q$. Let the group $\mathbb{Z}_s^*$ be the domain of set values. Due to the homomorphic properties of $\mathsf{Com}$, anyone can obliviously evaluate a commitment to $f(z)$ for any $z \in \mathbb{Z}_s^*$.

The HW construction uses this ability by having a verifier $\mathcal{V}$ compute a polynomial $f$ with $A$ as its set of roots. $\mathcal{P}$ can then obliviously evaluate $f$ and return the result to $\mathcal{V}$. Note, this is not a contribution due to HW. Similar constructions were proposed by Naor and Pinkus [90] and FNP [46]. It is also the basis of the KM scheme [71]. $\mathcal{V}$'s polynomial is constructed as shown in figure 4-4.

OBLIVIOUS POLYNOMIAL EVALUATION:

1. $\mathcal{V}$ chooses a random constant or irreducible polynomial $G(x)$.
2. $\mathcal{V}$ computes $f(x) = G(x) \cdot (\prod_{a_i \in A}(x - a_i)) = \sum_{i=0}^{|A|} \alpha_i x^i \in \mathbb{Z}_q[x]$.
3. If any $\alpha_i = 0$, restart the protocol.
4. $\mathcal{V}$ chooses a random polynomial $r(x) = \sum_{i=0}^{|A|} \gamma_i x^i \in \mathbb{Z}_p[x]$.
5. $\mathcal{V}$ publishes commitments $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i}$, for $i = 0$ to $|A|$.

Figure 4-4: HW oblivious polynomial evaluation

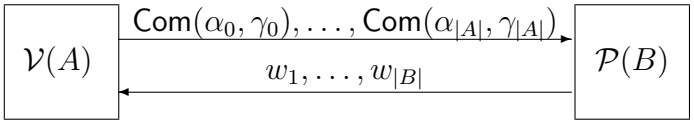


Figure 4-5: An illustration of HW private disjointness testing

Given these commitments to the $\alpha_i$ coefficients, $\mathcal{P}$ may use the homomorphic operations to compute a commitment to $f(z)$ for an arbitrary point $z \in \mathbb{Z}_s^*$:

$\prod_i \mathsf{Com}(\alpha_i, \gamma_i)^{z^i} = g^{\sum_i \alpha_i z^i} h^{\sum_i \gamma_i z^i} = g^{f(z)} h^{r(z)} = \mathsf{Com}(f(z), r(z))$

Because $\mathcal{P}$ does not want to accidentally reveal information about values $z \notin A$ to $\mathcal{V}$, he can select a random $R \in \mathbb{Z}_n^*$ and compute the value $\mathsf{Com}(Rf(z), Rr(z)) = g^{Rf(z)} h^{Rr(z)} = \mathsf{Com}(f(z), r(z))^R$. If $f(z) \neq 0 \bmod q$, then $Rf(z)$ will be some random value in $\mathbb{Z}_n$, and $\mathsf{Com}(f(z), r(z))^R$ will be some random value in $\mathbb{G}$.

However, if $f(z) = 0 \bmod q$, then $g^{Rf(z)}$ will have order $p$ (or 1). Since $h$ has order $p$, this means that $\mathsf{Com}(f(z), r(z))^R$ will have order $p$, which can be tested by $\mathcal{V}$ by checking if the $\mathsf{Test}$ operation returns a 1 value. Thus, if $\mathcal{P}$ returns some value with order $p$, $\mathcal{V}$ concludes that $\mathcal{P}$ obliviously evaluated the polynomial at a root.

Recall that $\mathcal{P}$ does not know $p$, $q$, or even $g$ or $h$. To erroneously convince $\mathcal{V}$ that he knows a root, a malicious $\mathcal{P}^*$ must produce some value of order $p$. Finding such a value is at least as hard as the Subgroup Computation Problem described in Definition 15.

## 4.4 HW Private Disjointness Testing

Given the oblivious polynomial evaluation protocol from the previous section, the HW construction to implement Private Disjointness Testing with a testable and homomorphic commitment primitive is quite simple. As mentioned, the overall protocol paradigm originally proposed by FNP [46]. Figure 4-5 illustrates the HW private disjointness testing protocol that is specified in figure 4-6.

HW Private Disjointness Testing:

1. $\mathcal{V}$ runs $S(1^k)$ to obtain $(\mathbb{G}, p, q)$, selects random generators $g, u$ in $\mathbb{G}$, and computes $n = pq$ and $h = u^q$. $\mathcal{V}$ publishes $(\mathbb{G}, n)$.
2. $\mathcal{V}$ and $\mathcal{P}$ announce $|A|$ and $|B|$ for respective input sets $A$ and $B$, which are poly$(k)$-sized subsets of $\mathbb{Z}_s^*$.
3. $\mathcal{V}$ publishes commitments to polynomial coefficients $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i} \in \mathbb{G}$ for $i = 0$ to $|A|$.
4. For each $b_j \in B$ selected in random order:
   (a) $\mathcal{P}$ obliviously evaluates $f(b_j)$ as $v_j = g^{f(b_j)} h^{r(b_j)}$.
   (b) $\mathcal{P}$ selects a random exponent $R_j \in \mathbb{Z}_n^*$.
   (c) $\mathcal{P}$ sends $\mathcal{V}$ the value $w_j = v_j^{R_j}$.
5. $\mathcal{V}$ halts if any $w_j = 1$.
6. $\mathcal{V}$ tests each $w_j$ by computing $w_j^p$.
7. If any $w_j^p = 1$, then $\mathcal{V}$ concludes that $A \cap B \neq \emptyset$.
8. Otherwise, $\mathcal{V}$ concludes $A \cap B = \emptyset$.

Figure 4-6: HW private disjointness testing

**Theorem 14** *The HW construction is correct and secure, i.e., it satisfies Definition 16, under the Subgroup Decision and the Subgroup Computation assumptions.*

**Remark:** Note that when talking to an honest prover, a verifier will actually learn $|A \cap B|$ in this protocol by counting the number of elements returned with order $p$. We could change the protocol to obfuscate this value, but having the prover return a random number of copies of each element in his set. This would not be true zero-knowledge, but it would be good enough for many practical applications. After all, there is no guarantee that the number of elements with order $p$ is the correct cardinality, because a malicious prover might evaluate the same value many times. This protocol can be modified to hide $|A \cap B|$ at a cost of increased communication as discussed in Section 4.6.3. In many PDT applications this extra information is not a problem, but users should still be aware that cardinality is revealed when provers are honest.

### 4.4.1 Security Proof

Theorem 14 is proven in four steps: completeness, soundness, malicious-prover zero knowledge, and honest-verifier zero knowledge.

**Proof of Completeness**

Recall the PDT completeness property:

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = D(A,B)\right] \geq (1 - \text{negl}(k))$$

In the oblivious polynomial evaluation protocol discussed in Section 4.3.3, a verifier $\mathcal{V}$ sends $(|A|+1)$ different $\mathsf{Com}(\alpha_i, \gamma_i)$ values to a prover $\mathcal{P}$, where $f(x) = \sum \alpha_i x^i$ and $r(x) = \sum \gamma_i x^i$. $\mathcal{P}$ will respond with $|B|$ values $w_j = \mathsf{Com}(f(b_j), r(b_j))^{R_j}$ for random $R_j$. If the protocol is followed correctly, then with high probability $b_j \in A$ if and only if $ord(w_j) = p$.

**Case 1:** $b_j \in A \implies ord(w_j) = p$ w.h.p. over $r(x) \in \mathbb{Z}_p[x]$.

If $b_j \in A$ then $f(b_j) = kq$ for some $k$. Then we have that the value $\mathsf{Com}(f(b_j), r(b_j))^{R_j} = (g^{kq}h^{r(b_j)})^{R_j}$. Recalling that $n = pq$, $g^n = 1$, and that $h^p = 1$, we have that the value $w_j^p = (g^{kn}h^{r(b_j)p})^{R_j} = 1$.

Note that there is also a negligible chance that $R_j kq = R_j r(b_j) = 0 \bmod p$, i.e. $w_j = 1$. This will cause $\mathcal{V}$ to halt the protocol. Since $R_j \in \mathbb{Z}_n^*$, this would mean that $k = r(b_j) = 0 \bmod p$.

Recall that $r(x)$ is chosen uniformly at random from $\mathbb{Z}_p[x]$. Thus, the chance that a particular $b_j$ is a root is at most $|A|/p$. The chance that none of the $|B|$ values are roots of $r(x)$ is:

$$\left(1 - \frac{|A|}{p}\right)^{|B|} = \left(\left(1 - \frac{1}{\frac{p}{|A|}}\right)^{\frac{p}{|A|}}\right)^{\frac{|A||B|}{p}} \approx e^{-\left(\frac{|A||B|}{p}\right)} > e^{-\left(\frac{\text{poly}(k)}{2^k}\right)} > 1 - \text{negl}(k)$$

**Case 2:** $b_j \notin A \implies ord(w_j) \neq p$.

If $b_j \notin A$, then $f(b_j) \neq 0 \bmod q$. So, $w_j^p = g^{R_j f(b_j) p} h^{R_j r(b_j) p} = g^{R_j f(b_j) p}$. Note that $R_j \in \mathbb{Z}_n^*$, so $R_j f(b_j) \neq 0 \bmod q$. Therefore, we have $R_j f(b_j) p \neq 0 \bmod n$ and can conclude that $w_j^p \neq 1$.

**Proof of Soundness**

Recall the PDT soundness property:

$$\forall \mathcal{P}^*_{ppt},\ \Pr_{A \in U}\left[\mathcal{P}^*\mathcal{V}(A) \neq 0\right] \leq \text{negl}(k)$$

The verifier $\mathcal{V}$ will only accept when one of the poly$(k)$ values sent to $\mathcal{V}$ by $\mathcal{P}^*$ has order $p$. Recall that (even a malicious) $\mathcal{P}^*$ only knows $(\mathbb{G}, n)$ and $\mathcal{V}$'s commitments. It does not know $p$, $q$, $h$ or even $g$.

In the given soundness definition, $\mathcal{V}$ is operating with a *random* set $A \in U$. $\mathcal{P}^*$ has no *a priori* knowledge about $A$, other than $|A|$. By testable and homomorphic

commitment hiding property, $\mathcal{P}^*$ can learn nothing about $A$ from the commitments themselves.

It is worth elaborating on this particular definition of soundness. It assumes that a malicious prover knows nothing about the set he is trying to cause a spurious intersection with. Otherwise, if he has some partial knowledge, say that some element is in the set $A$ with a 1/1000 chance, a malicious prover could cause a spurious intersection with with 1/1000 probability. Our formulation captures the notion that malicious provers should not be able to trick verifiers that an intersection exists without the use of some previous knowledge of $A$.

Alternatively, one could formulate this as an experiment where an adversary $\mathcal{P}^*$ chooses a set $B^*$ as input for an honest PDT prover $\mathcal{P}$. Any partial knowledge of $A$ could be embedded in $B^*$. The probability that $\mathcal{V}$ believes there is an intersection interacting with $\mathcal{P}(B^*)$ would be non-negligibly different than when interacting directly with $\mathcal{P}^*$. However, the soundness formulation as given is clearer and captures the same properties.

With no information on $A$, $\mathcal{P}^*$ can try to evaluate $f(x)$ at $|B^*| = \text{poly}(k)$ random values and will fail to guess a member of $A$ with probability approximately $e^{(-|A||B^*|/p)}$ which is greater than $1 - \text{negl}(k)$. Note by the Subgroup Decision Assumption, $\mathcal{P}^*$ won't actually be able to verify when he correctly guesses a value in $A$.

There is one caveat concerning the distribution of $\alpha$ coefficients. It could be the case that some coefficient, or linear combination of coefficients, has a non-negligible chance of being zero. Note that a zero coefficient corresponds to a commitment of the form $\mathsf{Com}(0, r_i) = g^0 h^{r_i}$. In other words, the commitment corresponding to a zero coefficient has order $p$. Thus, all $\mathcal{P}^*$ would have to do to break soundness is return this commitment to $\mathcal{V}$. The same applies if some linear combination of coefficients is zero with non-negligible probability.

To avoid this issue, $\alpha_i$ values are checked when $f(x)$ is created to ensure they are non-zero. Recall that to generate $f(x)$ the verifier will choose a random constant or irreducible polynomial $G(x)$ and multiply it by $\prod(x - a_i)$. If one $G(x)$ fails, another random irreducible polynomial can be chosen until all $\alpha_i$ values are non-zero. For each iteration, there is a high probability that no coefficients will be zero, so with high probability a constant number of iterations will be necessary.

Since the $\alpha_i$ coefficients are determined by some random irreducible polynomial $G(x)$, and in this case, a random set $A$, they are unpredictable to a prover $\mathcal{P}^*$. A malicious prover cannot send any trivial linear combinations of committed coefficients back to the verifier since the coefficients are determined entirely by a random $A$ and $G(x)$.

Thus, $\mathcal{P}^*$ essentially must try to generate order $p$ values directly from $(\mathbb{G}, n)$. We will use the Subgroup Computation Assumption (SCA), from Definition 15. The SCA asserts that it is difficult for a polynomial-time adversary, given the description of an efficiently sample-able group $\mathbb{G}$ of order $n = pq$, to find an element of order $p$. We will show that any adversary $\mathcal{A}$ that violates Soundness also violates the SCA.

Suppose we have some adversary $\mathcal{A}$ that violates Soundness. $\mathcal{A}$ would take $(\mathbb{G}, n)$ and $\mathcal{V}$'s commitments as input and would have a non-negligible probability of returning an element of order $p$. By the MPZK property, $\mathcal{A}$'s behavior cannot change

significantly when we substitute $\mathcal{V}$'s commitments with random values.

Given such an adversary $\mathcal{A}$ and setup $(\mathbb{G}, n)$, we could feed it random values sampled from $\mathbb{G}$ and obtain an element of order $p$. Obviously, this violates the SCA. Thus by the facts that $A$ is chosen uniformly at random, that $f$ has a negligible probability of having zero coefficients, and by the SCA, the Soundness property from the PDT definition holds.

**Proof of PDT Malicious-Prover Zero Knowledge**

Recall the PDT Malicious-Prover Zero Knowledge (MPZK) property:

$$\exists \mathcal{S}_{ppt},\ \forall \mathcal{P}^*_{ppt},\ \forall A \in U,\ \{\mathsf{View}^{\mathcal{P}^*}\left[\mathcal{P}^*\mathcal{V}(A)\right]\} \overset{c}{\approx} \{\mathsf{View}^{\mathcal{P}^*}\left[\mathcal{P}^*\mathcal{S}(1^{|A|})\right]\}$$

The proof that a malicious-prover $\mathcal{P}^*$ is not able to glean any information about $A$ from the values $\mathsf{Com}(\alpha_i, \gamma_i)$ follows from the hiding property of the testable and homomorphic commitment scheme from Lemma 4 (which in turn essentially follows from the semantic security of the BGN cryptosystem [13].) The proof will argue this step, then describe a zero-knowledge simulator.

First, the Subgroup Decision Assumption (SDA), from Definition 14, implies that $\mathsf{Com}$ is computationally hiding. Suppose the contrapositive, that $\mathsf{Com}$ is not hiding. Let $\mathcal{A}$ be a probabilistic polynomial time adversary that runs the indistinguishability under chosen plaintext attack (IND-CPA) experiment shown in figure 4-7 for a given input $(\mathbb{G}, n)$ where $g, h \in \mathbb{G}$ and $h$ has order $p$.

$\text{Exp}^{IND-CPA}_{\mathcal{A}}(\mathbb{G}, n, g, h)$:

1. $\mathcal{A}(\mathbb{G}, n) \rightarrow (a_0, a_1)$
2. $t \leftarrow \mathbb{Z}^*_n$
3. $b \leftarrow \{0, 1\}$
4. $\mathcal{A}(\mathbb{G}, n, g^{a_b} h^t) \rightarrow b'$

Figure 4-7: Testable and homomorphic commitment IND-CPA experiment

Suppose, for the sake of contradiction, that $\Pr[b = b'] \geq 1/2 + 1/\text{poly}(k)$. If this is the case, we can construct an adversary $\mathcal{A}^*$ that violates the SDA assumption. To do so, given same input $(\mathbb{G}, n)$ and a challenge $x \in \mathbb{G}$, the adversary $\mathcal{A}^*$ must be able to distinguish whether $x = g^r$ or $g^{rq}$ for $r \in \mathbb{Z}^*_n$. (We ignore the case where $x = 1$.)

Now, $\mathcal{A}^*$ selects a random generator $g \in \mathbb{G}$ and runs $\text{Exp}^{IND-CPA}_{\mathcal{A}}(\mathbb{G}, n, g, x)$ with the adversary $\mathcal{A}$. If $x = g^{rq}$ then it will have order $p$, and thus the $\mathcal{A}^*$ simulates $\text{Exp}^{IND-CPA}_{\mathcal{A}}$ perfectly for $\mathcal{A}$, so $\mathcal{A}$ will maintain a $1/2 + 1/\text{poly}(k)$ advantage.

If $x = g^r$, in step 4 of $\text{Exp}^{IND-CPA}_{\mathcal{A}}$, the adversary $\mathcal{A}$ will receive the value $g^{a_0} g^{rt}$ (w.l.o.g.). Because $r$ and $t$ are chosen uniformly at random from $\mathbb{Z}^*_n$, it's equally

possible that $\mathcal{A}$ has received the value $g^{a_1}g^{r't'}$, where $r't' = rt + a_0 - a_1$. Thus, $\mathcal{A}$ has necessarily a $1/2$ chance of guessing $b$ when $x = g^r$.

By running repeated $\text{Exp}_{\mathcal{A}}^{IND-CPA}(\mathbb{G}, n, g, x)$ experiments, $\mathcal{A}^*$ can observe $\mathcal{A}$'s performance at guessing $b$. If $\mathcal{A}$ displays a $1/2 + 1/\text{poly}(k)$ advantage, $\mathcal{A}^*$ will guess that $x = g^{rq}$. If $\mathcal{A}$ has no advantage, then $\mathcal{A}^*$ will guess that $x = g^r$. Thus, $\mathcal{A}^*$ can distinguish distributions of $(\mathbb{G}, n, g^{rq})$ from $(\mathbb{G}, n, g^r)$, violating the SDA.

Therefore, if the SDA holds, then the commitment scheme $\mathsf{Com}$ is computationally hiding. A malicious adversary cannot distinguish a commitment to a particular $a_0$ from a commitment to a random message. Based on this, we will construct a simulator $\mathcal{S}(1^{|A|})$ that will be indistinguishable from a verifier $\mathcal{V}(A)$ to any probabilistic polynomial time $\mathcal{P}^*$.

$\mathcal{S}$ is quite simple: it will send $|A|$ random values in $\mathbb{G}$ to $\mathcal{P}^*$. Because random values are individually indistinguishable from commitments to particular $\alpha_i$ coefficients, $\mathcal{P}^*$ will not be able to distinguish $\mathcal{S}(1^{|A|})$ from $\mathcal{V}(A)$. Thus, by the SDA, the Malicious-Prover Zero Knowledge property holds.

**Proof of Honest-Verifier Perfect Zero Knowledge**

Recall the PDT Honest-Verifier Perfect Zero Knowledge (HVPZK) property:

$$\exists \mathcal{S}_{ppt},\ \forall A \in U,\ \forall B \in U,\ \{\mathsf{View}^{\mathcal{V}}\big[\mathcal{P}(B)\mathcal{V}(A)\big]\} \approx \{\mathcal{S}(A, 1^{|B|}, 1^{|A\cap B|})\}$$

The HVPZK property implies that an adversary given $\mathsf{View}^{\mathcal{V}}[\mathcal{P}(B)\mathcal{V}(A)]$ cannot learn anything about $B$ that it could not learn given $|A \cap B|$ and $|B|$. The quantification is over all choices of input $A$, which includes adversarial choices of $A$ that might be based on some prior knowledge or some partial knowledge of a particular set $B$. By running the legitimate protocol, however, a semi-honest $\mathcal{V}$ should not learn anything beyond what is can conclude from the size of $|A \cap B|$.

Because the adversary is semi-honest, it cannot deviate from the protocol, manipulate its choice of $(\mathbb{G}, n)$, or manipulate its committed coefficients. Its power is equivalent to choosing a query set $A$ and running the legitimate verifier $\mathcal{V}$. We will describe a simulator $\mathcal{S}$, that on inputs $(A, 1^{|B|}, 1^{|A\cap B|})$ produces a view that is perfectly indistinguishable from $\mathsf{View}^{\mathcal{V}}[\mathcal{P}(B)\mathcal{V}(A)]$.

The HVPZK simulator works as shown in figure 4-8 (recall the PDT protocol from Section 4.4). This simulator perfectly generates the distribution $\mathsf{View}^{\mathcal{P}}[\mathcal{P}(B)\mathcal{V}(A)]$. Here, $\mathcal{S}$ follows the same setup as $\mathcal{V}$ and $\mathcal{P}$ in steps 1, 2, and 3. In step 4, the response of the simulated prover is $|A \cap B|$ *random* values of order $p$ in $\mathbb{G}$ and $|B| - |A \cap B|$ *random* values in $\mathbb{G}$. This is exactly the distribution than an honest prover would return. Thus, the two views are perfectly indistinguishable and the HVPZK property holds.

## 4.5 Semi-Honest Private Intersection Cardinality

The construction in Section 4.3 is *not* an Honest-Verifier Private Intersection Cardinality protocol. Unfortunately, there are trivial ways a malicious-prover can manipulate the actual cardinality value obtained by the verifier. The simplest attack

HVPZK Simulator:

1. $\mathcal{S}$ runs $S(1^k)$ to obtain $(\mathbb{G}, p, q)$, selects random generators $g, u$ in $\mathbb{G}$, and computes $n = pq$ and $h = u^q$. $\mathcal{S}$ publishes $(\mathbb{G}, n)$.
2. $\mathcal{S}$ announces $|A|$ and $|B|$.
3. $\mathcal{S}$ publishes committed polynomial coefficients $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i}$ in $\mathbb{G}$ for $i = 0$ to $|A|$, exactly as $\mathcal{V}$ would.
4. $\mathcal{S}$ ignores the values from step 3 and generates $|B|$ random elements of $\mathbb{G}$, raises a random selection of $|A \cap B|$ of these values to $q$ (thereby making them of order $p$), leaves the other $|B| - |A \cap B|$ values as is, and outputs these $|B|$ elements as the response of $\mathcal{P}$.

Figure 4-8: Honest-verifier perfect zero knowledge simulator

would be to obliviously evaluate each element in $B$ twice. The verifier will think the cardinality is $2 \cdot |A \cap B|$. By the HVPZK property, an honest verifier cannot detect this attack, otherwise it could distinguish different evaluations by the prover.

For this reason, the HW construction violates the Cardinality Soundness property from definition 17. However, we may consider a weaker PIC setting by assuming that both the prover and verifier are honest-but-curious (semi-honest). Recall that a honest-but-curious party will follow a protocol as specified, but may further examine any received values with the intention of learning more [51].

**Definition 19 (Semi-Honest Private Intersection Cardinality)** *An* Semi-Honest Intersection Cardinality *protocol has the same setup as in Definition 16, except for the following difference:*

**Completeness:** *For semi-honest parties, the protocol works and the verifier learns the cardinality predicate; that is,*

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = |A \cap B|\right] \geq (1 - \mathrm{negl}(k))$$

*where probability is taken over the randomness of $\mathcal{P}$ and $\mathcal{V}$.*

**Corollary 1** *The HW construction from Section 4.3 implements a Semi-honest Private Intersection Cardinality Protocol, under the Subgroup Decision and the Subgroup Computation assumptions.*

Corollary 1 follows directly from the proof of Theorem 14.

## 4.6 Discussion

### 4.6.1 Malicious Verifiers

The HW construction is only secure against honest-but-curious verifiers. A malicious verifier $\mathcal{V}^*$ can choose arbitrary setup parameters $(\mathbb{G}, n)$, such as $\mathbb{G} = \mathbb{Z}_{p'}$ where $p' = 2n+1$, and send $\mathcal{P}$ an arbitrary set of values $g^{c_i} \in \mathbb{G}$, where the $c_i$ values define some polynomial $f(x) = \sum c_i x^i$. In response, a legitimate $\mathcal{P}$ will send values $w = g^{Rf(b)}$ for each $b \in B$, where $R$ is chosen uniformly at random from $\mathbb{Z}_n^*$.

If $g^{f(b)}$ has order $n$, then $w$ will be a random element of order $n$. However, a malicious $\mathcal{V}^*$ can design the polynomial $f(\cdot)$ to have different orders for different inputs. So, if $p' = 2pq + 1$, $\mathcal{V}^*$ might have two sets $S, T$ such that $\forall s \in S, f(s) = 0 \bmod p$ and $\forall t \in T, f(t) = 0 \bmod q$. Thus, $\mathcal{V}^*$ would be able to distinguish how many elements of $B$ were in either $S$ or $T$. In fact, $\mathcal{V}^*$ could choose $n$ to have many factors. This would allow her to test how many elements of $B$ belonged to any of several different sets.

To make the HW construction secure against malicious verifiers, $\mathcal{V}$ could provide a zero knowledge proof that $n$ was the product of two large primes $p$ and $q$. $\mathcal{V}$ could then include a proof that each of her commitments was the product of at least one value with order $p$. Camenisch and Michels describe efficient zero knowledge proofs which can be applicable in this setting [17]. Of course, the costs of creating and verifying these proofs may be equivalent to the costs of the existing malicious verifier-secure protocols due to FNP and KM.

### 4.6.2 Computation and Communication Costs

The computation and communication costs of the HW construction are equivalent to the costs of FNP's malicious-prover secure scheme, except the HW construction offers security against malicious provers without random oracles. The costs of HW are as follows:

**$\mathcal{V}$ Computation Costs:** Computing $\alpha_i$ coefficients naively requires $O(|A|^2)$ modular additions and multiplications. Committing requires $O(|A|)$ modular exponentiations and multiplications. Testing whether responses have order $p$ requires $O(|B|)$ modular exponentiations.

**$\mathcal{P}$ Computation Costs:** Using Horner's method, $\mathcal{P}$ can obliviously evaluate a $d$-degree polynomial with $O(d)$ modular exponentiations and multiplications. Normally, $\mathcal{P}$ will perform $O(|A||B|)$ operations; that is, one polynomial evaluation at a cost of $O(|A|)$ operations for each of the $|B|$ elements in $\mathcal{P}$'s set. However, as described in FNP, if the balanced hash-bucket scheme of Azar et al. [3] is employed $\mathcal{P}$ can perform only $O(|B| \ln \ln |A|)$ modular operations.

**Communication Costs:** The total exchange between $\mathcal{P}$ and $\mathcal{V}$ is $O(k(|A|+|B|))$ bits or $O(k(|A| \ln \ln |A| + |B|))$ if a hash-bucket optimization is used, where $1^k$ is the security parameter.

### 4.6.3 Hiding Set Sizes and Small Set Domains

In the HW construction, the size of the prover and verifier's sets is public information. In practice, however, the prover $\mathcal{P}$ with set $B$ or the verifier $\mathcal{V}$ with set $A$ might wish to mask the true size of their sets using well-known techniques. To do this, the verifier $\mathcal{V}$ can compute a random polynomial $f(\cdot)$ with roots in set $A$ as normal, then multiply it by some irreducible polynomial of arbitrary degree $d$. Then, $\mathcal{P}$ (or anyone else) will only learn that $\mathcal{V}$'s set is of some size less or equal to $|A| + d$. Similarly, $\mathcal{P}$ can evaluate $f$ on each value in $B$ an arbitrary number of times. Each copy will be randomized by the regular protocol. This will maintain correctness of Private Disjointness Testing, but would obviously change the results of an honest-but-curious private intersection cardinality protocol, as described in Section 4.5.

The HW construction requires that sets $A$ and $B$ are small with respect to the domain of set values. Obviously, in the HW PDT protocol, if $|B| = \Theta(\sqrt{n})$, then a malicious adversary can factor $n$ in time polynomial to the size of its input. This would allow an adversary to generate values of order $p$ and violate the Soundness property.

### 4.6.4 Private Information Retrieval

Recalling Private Information Retrieval (PIR), one party will have a database of $m+1$ bits $x_0, \ldots, x_m$, while a second party wishes to privately query a particular bit $x_i$ without revealing $i$. Putting this in the context of the HW construction, $A$ would be the set of indices where $x$ is 1 and $B = \{i\}$. Unfortunately, it may be the case that $|A|$ is large with respect to the domain $\mathbb{Z}_m^*$.

As a result, the requirement of small set domains mentioned in Section 4.6.3 precludes directly using the HW construction for PIR in general. Yamamura and Saito offer a simple PIR solution based on the SDA [124]. However, their PIR solution approach is very inefficient and requires $O(km)$ bits of communication to privately retrieve a single bit from a $m$-bit database, where $k$ is a security parameter.

### 4.6.5 Multiparty Extensions

Another interesting variant to the 2-party PDT protocol is considering a multi-verifier, single-prover PDT scenario. For example, suppose that law enforcement agencies from different countries, in the role of verifiers, wish to be assured by an airline, in the role of the prover, that no one on *any* of their watch-lists is getting on the next flight. The law enforcement agencies neither trust each other nor the airline with their individual databases, yet may want to corroborate their watch lists (so as to possibly work together).

Suppose there are two verifiers. The HW construction may be extended as follows. First, each verifier computes his own values $n_i = p_i q_i$ and a group of known order $\prod_i n_i$ is published. Next, both verifiers publish commitments to their own polynomials using a random generator $g$ from the group of order $n_1 n_2$ and, respectively, $h_1$ of order $(n_1 n_2)/p_1 = q_1 n_2$ and $h_2$ order $(n_1 n_2)/p_2 = n_1 q_2$. That is, values of

the form $g^{\alpha_i} h_1^{r_i}$ and $g^{\beta_j} h_2^{r_j}$, where $f(x) = \sum \alpha_i x^i$ and $z(x) = \sum \beta_j x^j$. A third party can obliviously evaluate commitments to the sum of these polynomials. If the third party's set contains an element $c_i$ such that $f(c_i) = z(c_i) = 0$, then this party can output elements $h_1^r h_2^{r'}$, which have order $q_1 q_2$.

This is interesting because no single party could compute elements of order $q_1 q_2$ by themselves; this only occurs when the airline makes an evaluation on an element contained in *both* of the law enforcement agencies' sets. Each agency, knowing $q_1$ and $q_2$ respectively, could collaborate to detect this fact and take further action. The benefit here is that the contents of the sets of the law enforcement agencies and the airline all remain private, up to knowledge of any three-way intersections. This digression is just to illustrate that unknown order subgroups might be applied in other interesting applications.

### 4.6.6 Finding Intersection Values with HW

As previously mentioned, basic FNP is actually a Private Intersection or Private Matching protocol. The verifier party learns which specific values are in the set intersection. Essentially, the prover will send homomorphic encryptions of the form $E_{pk}(r \cdot f(b) + b)$ for values $b \in B$. If $b \in A$, then $f(b) = 0$ and the verifier will receive an encryption of $b$. Otherwise, the verifier receives a random value.

Of course, this is still susceptible to malicious prover attacks. A malicious prover can encrypt any value he likes or can encrypt values like $E_{pk}(r_1 \cdot f(b_1) + r_2 \cdot f(b_2) + b_1)$, which can be interpreted as "If $(b_1 \in A)$ and $(b_2 \in A)$, then tell the verifier that $(b_1 \in A)$". FNP's fixes the problem by using the random oracle model to force a prover to use the encrypted coefficient values prepared by the verifier.

This begs the question of whether the HW testable and homomorphic commitment primitive could be used in a private intersection protocol. Initially, one may consider using the exact FNP construction and having the prover obliviously evaluate $g^{Rf(b)+b} h^r$. If $f(b) = 0$, raising this to the power $q$ will result in the value $(g^q)^b$. The verifier can then check whether for any of its own values $a$, that $(g^q)^a = (g^q)^b$.

Unfortunately, like FNP, a malicious prover could also send conditional evaluations, like "if x is in A, then reveal that y is in B". This would violate the soundness of a private intersection protocol. Thus, a HW-style private intersection protocol offers no advantage over FNP. They have equivalent computation costs and the same level of security.

An open question is whether a HW testable and homomorphic commitment-based private intersection protocol may be constructed without the use of random oracles. It may be possible to leverage groups that have several different ordered subgroups as discussed in Section 4.6.5. The verifier might be able to commit $A$ in a polynomial $f$ that has roots in different subgroups. Then a malicious prover would not be able simply to mix-and-match oblivious evaluations of different $b$ values. This idea is not fully developed and can be the subject of future work.

## 4.7 Conclusion and Open Questions

This chapter presented the Hohenberger-Weis (HW) honest-verifier private disjointness testing protocol, which originally appeared in [59]. HW is secure against malicious provers without requiring multiple invocations, random oracles, non-interactive zero knowledge proofs, or universally-composable commitments. The related FNP and KM protocols require one or more of these assumptions to be made secure against malicious provers, while HW requires only the subgroup decision and subgroup computation assumptions.

There are several open questions and problems related to HW. First, disjointness testing is a fairly limited application. An open question is whether there are natural constructions of private set operations like union or intersection based on the subgroup assumptions. It is likely that several of the FNP-inspired privacy-preserving set operations due to Kissner and Song [72] may be adapted using this chapter's testable and homomorphic commitment primitive.

The testable and homomorphic commitment based on the subgroup assumptions and described in Section 4.3.2 may be useful in other applications. Essentially, this commitment scheme allows one party to obliviously evaluate functions, and another party to test properties of the evaluation. In this chapter there was a single property that was tested – whether a polynomial evaluated to zero.

However, testable and homomorphic commitment may be especially useful with groups with many subgroups of unknown order, as described in Section 4.6.5. This would allow a party to test several properties of an evaluation, or even several parties to test different properties independently. As discussed at the end of Section 4.6.6, groups with many subgroups of different order might be useful in developing a HW testable and homomorphic commitment-based private intersection protocol – where the verifier learns the actual values of the intersection.

More research must be focused on the hardness of both the subgroup decision and subgroup computation assumptions. First, does one assumption imply the other? Second, what is the relation, if any, between these assumptions and the Decisional or Computation Diffe-Hellman assumptions? Either subgroup assumption implies that factoring is hard, otherwise someone could just factor a group's order to obtain the order of its subgroups. Does assuming that factoring is hard imply either of the subgroup decision assumptions? These are all important questions that are relevant to both the HW construction and other works based on subgroup decision assumptions.

# Appendix A

# Glossary

**BGN** Boneh, Goh, Nissim encryption [13]; hardness based on SCA, page 88.

**BKW** Blum, Kalai, and Wasserman [11] algorithm for solving the LPN problem; best known asymptotic runtime, page 43.

**Canonical** SRS property; both confluent and terminating, page 49.

**CIND-CPA** Cascadable indistinguishability under chosen plaintext attack, page 54

**Confluence** SRS property; any string's derivations must share a common derivation, page 49.

**$\mathsf{d}$-skip rules** Rewrite rules in $\mathcal{S}_{com}$ of the form $\mathsf{d}_k\mathsf{e}_j \rightarrow \mathsf{e}_j\mathsf{d}_k$, sufficient to model commutative properties, page 64.

**Efficiently Cascadable** Property of cascadable cryptosystems; length of ciphertexts will grow linearly as a function of the net encryption height, page 53.

**Equivalence** SRS property; strings are equivalent if they have the same canonical form, page 49.

**EPC** Electronic product codes, page 20.

**FNP** Freedman, Nissim, and Pinkas [46] PIC protocol, 80.

**GJJS** Golle, Jakobsson, Juels and Syverson [53] universal re-encryption scheme, page 71.

**HB** Hopper-Blum human-to-computer authentication protocol, page 24.

**HB+** Augmented Hopper-Blum tag-to-reader authentication protocol, page 28.

**HW** Hohenberger and Weis [59], honest-verifier PDT protocol, page 88.

**HVPZK** Honest-verifier perfect zero knowledge, page 86.

**IND-CCA** Indistinguishability under adaptive chosen ciphertext attack, 60

**IND-CPA** Indistinguishability under chosen plaintext attack, page 54

**IND-RCCA** Indistinguishability under re-playable adaptive chosen ciphertext attack, 60

**KM** Kiayias and Mitrofanova [71] PDT protocol, page 80.

**LPN** Learning Parity with Noise problem, page 26.

**MPZK** Malicious-prover zero knowledge, page 86.

**Net Encryption Height** Given a SRS string $s$, minimum length of a string $t$ such that $ts \simeq \epsilon$, page 51.

**PDT** Private disjointness testing; two parties wish to determine whether their respective sets contain any mutual elements, page 79.

**PIC** Private intersection cardinality; two parties wish to compute the value of the intersection of their two private sets, page 79.

**Rewrite Fidelity** Notion that evaluating a SRS string will be equivalent to evaluating its canonical form, page 52.

**RFID** Radio frequency identification, page 19.

**SCA** Subgroup computation assumption, page 85.

**SDA** Subgroup decision assumption, page 85.

**SRS** String rewrite system, page 49.

**SVP** Shortest vector problem, page 26.

**Termination** SRS property; only a finite number of rules may be applied to any string, page 49.

**THC** Testable and homomorphic commitment, page 89.

# Bibliography


[1] Agrawal, R., Evfimievski, A., and Srikant, R. Information sharing across private databases. In *International Conference on Management of Data (ACM SIGMOD)* (2003), ACM Press, pp. 86–87.

[2] Anderson, R., and Kuhn, M. Low cost attacks on tamper resistant devices. In *International Workshop on Security Protocols* (1997), B. Christianson, B. Crispo, T. M. A. Lomas, and M. Roe, Eds., vol. 1361 of *Lecture Notes in Computer Science*, Springer, pp. 125–136.

[3] Azar, Y., Broder, A. Z., Karlin, A. R., and Upfal, E. Balanced allocations. *SIAM Journal on Computing 29*, 1 (1999), 180–200.

[4] Barker, W. C. Recommendation for the triple data encryption algorithm (TDEA) block cipher. Tech. Rep. 800-67, National Institute of Standards and Technology, 2004.

[5] Bellare, M., and Rogaway, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Computer and Communications Security – CCS* (1993), ACM Press, pp. 62–73.

[6] Bellare, M., and Rogaway, P. Optimal asymmetric encryption – How to encrypt with RSA. In *Advances in Cryptology – EUROCRYPT '94* (1994), A. D. Santis, Ed., vol. 950 of *Lecture Notes in Computer Science*, Springer, pp. 92–111.

[7] Bellido, M., Acosta, A., Valencia, M., Barriga, A., and J.L.Huertas. Simple binary random number generator. *Electronic Letters 28*, 7 (March 1992), 617–618.

[8] Berlekamp, E. R., McEliece, R. J., and Van Tilborg, H. C. A. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory 24*, 3 (May 1978), 384–386.

[9] Bing, B. *Broadband Wireless Access*. Kluwer Academic Publishers, 2002.

[10] Blum, A., Furst, M., Kearns, M., and Lipton, R. J. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology – CRYPTO'93* (1993), D. R. Stinson, Ed., vol. 773 of *Lecture Notes in Computer Science*, Springer, pp. 278–291.

[11] Blum, A., Kalai, A., and Wasserman, H. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM 50*, 4 (July 2003), 506–519.

[12] Blum, M., Luby, M., and Rubinfeld, R. Self-testing/correcting with applications to numerical problems. In *Symposium on Theory of Computation – STOC* (1990), ACM Press, pp. 73–83.

[13] Boneh, D., Goh, E.-J., and Nissim, K. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography (TCC)* (2005), J. Kilian, Ed., vol. 3378 of *Lecture Notes in Computer Science*, Springer, pp. 325–341.

[14] Bono, S., Green, M., Stubblefield, A., Juels, A., Rubin, A., and Szydlo, M. Security analysis of a cryptographically-enabled RFID device. In *USENIX Security Symposium* (July 2005), pp. 1–16.

[15] Bringer, J., Chabanne, H., and Dottax, E. HB++: A lightweight authentication protocol secure against some attacks. In *Security, Privacy, and Trust in Pervasive and Ubiquitous Computing – SecPerU,* (To appear June 2006).

[16] Bucci, M., Germani, L., Luzzi, R., Trifiletti, A., and Varanonuovo, M. A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC. *IEEE Transactions on Computers 52*, 4 (April 2003), 403–409.

[17] Camenisch, J., and Michels, M. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology – EUROCRYPT'99* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 107–122.

[18] Camenisch, J., and Shoup, V. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology – CRYPTO '03* (2003), D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, Springer, pp. 126–144.

[19] Canetti, R., and Fischlin, M. Universally composable commitments. In *Advances in Cryptology – CRYPTO '01* (2001), J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, Springer, pp. 19–40.

[20] Canetti, R., Goldreich, O., and Halevi, S. The random oracle methodology, revisited. *Journal of the ACM 51*, 4 (July 2004), 557–594.

[21] Canetti, R., Krawczyk, H., and Nielsen, J. B. Relaxing chosen-ciphertext security. In *Advances in Cryptology – CRYPTO '03* (2003), D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, Springer, pp. 565–583.

[22] Canetti, R., Lindell, Y., Ostrovsky, R., and Sahai, A. Universally composable two-party and multi-party secure computation. In *Symposium on Theory of Computation – STOC* (2002), J. H. Reif, Ed., ACM Press, pp. 495–503.

[23] Catalano, D., Gennaro, R., Howgrave-Graham, N., and Nguyen, P. Q. Paillier's cryptosystem revisited. In *Computer and Communications Security – CCS* (2001), ACM Press, pp. 206–214.

[24] Chabaud, F. On the security of some cryptosystems based on error-correcting codes. In *Advances in Cryptology – EUROCRYPT '95* (1994), A. D. Santis, Ed., vol. 950 of *Lecture Notes in Computer Science*, Springer, pp. 131–139.

[25] Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 4*, 2 (February 1981), 84–90.

[26] Chaum, D. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy 2*, 1 (January-February 2004), 38–47.

[27] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M. Y. Tools for privacy preserving distributed data mining. *SIGKDD Explorations 4*, 2 (January 2003), 1–7.

[28] Courtois, N., Finiasz, M., and Sendrier, N. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology – ASIACRYPT '01* (2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*, Springer, pp. 157–174.

[29] Cramer, R., Gennaro, R., and Schoenmakers, B. A secure and optimally efficient multi- authority election scheme. In *Advances in Cryptology – EUROCRYPT '97* (1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer, pp. 103–118.

[30] Crawford, J. M., Kearns, M. J., and Shapire, R. E. The minimal disagreement parity problem as a hard satisfiability problem. Computational Intelligence Research Laboratory and AT&T Bell Labs. Available at http://www.cs.cornell.edu/selman/docs/crawford-parity.pdf, February 1994.

[31] Damgård, I., and Nielsen, J. B. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Advances in Cryptology – CRYPTO '02* (2002), M. Yung, Ed., vol. 2442 of *Lecture Notes in Computer Science*, Springer, pp. 581–596.

[32] Dershowitz, N. A taste of rewriting. In *Functional Programming, Concurrency, Simulation and Automated Reasoning* (1993), P. E. Lauer, Ed., vol. 693 of *Lecture Notes in Computer Science*, Springer, pp. 199–228.

[33] Dershowitz, N., and Jouannaud, J.-P. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. Elsevier and MIT Press, 1990, pp. 243–320.

[34] Dershowitz, N., and Plaisted, D. A. *Handbook of Automated Reasoning*, vol. 1. Elsevier, 2001, ch. Rewriting, pp. 535–610.

[35] Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory 22*, 6 (November 1976), 644–654.

[36] Dodis, Y., and Katz, J. Chosen-ciphertext security of multiple encryption. In *Theory of Cryptography Conference (TCC)* (2005), J. Kilian, Ed., vol. 3378 of *Lecture Notes in Computer Science*, Springer, pp. 188–209.

[37] Dolev, D., Even, S., and Karp, R. M. On the security of ping-pong protocols. *Information and Control 55*, 1–3 (October/November/December 1982), 57–67.

[38] Dolev, D., and Yao, A. C. On the security of public key protocols. *IEEE Transactions on Information Theory 29*, 2 (March 1983), 198–208.

[39] EPCGlobal. *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz*, version 1.0.9 ed., 2004.

[40] EPCglobal. Website. http://www.epcglobalinc.org/, 2005.

[41] Federal Information Processing Standards (FIPS). Data encryption standard. Tech. Rep. 46-2, National Institute of Standards and Technology (NIST), 1988, supercedes FIPS 46-1.

[42] Federal Information Processing Standards (FIPS). Standard hash algorithm (SHA-1). Tech. Rep. 180-1, National Institute of Standards and Technology (NIST), 1995, supercedes FIPS 180.

[43] Feldhofer, M., Dominikus, S., and Wolkerstorfer, J. Strong authentication for RFID systems using the AES algorithm. In *Cryptographic Hardware and Embedded Systems – CHES* (2004), M. Joye and J.-J. Quisquater, Eds., vol. 3156 of *Lecture Notes in Computer Science*, Springer, pp. 357–370.

[44] Floerkemeier, C., and Lampe, M. Issues with rfid usage in ubiquitous computing applications. In *Pervasive Computing – PERVASIVE* (2004), A. Ferscha and F. Mattern, Eds., vol. 3001 of *Lecture Notes in Computer Science*, Springer, pp. 188–193.

[45] Food and Drug Administration. Combating counterfeit drugs. Tech. rep., US Department of Health and Human Services, Rockville, Maryland, Februrary 2004.

[46] Freedman, M. J., Nissim, K., and Pinkas, B. Efficient private matching and set intersection. In *Advances in Cryptology – EUROCRYPT '04* (May 2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 1–19.

[47] Fujisaki, E., and Okamoto, T. How to enhance the security of public-key encryption at minimum cost. In *Public-Key Cryptography* (1999), H. Imai and Y. Zheng, Eds., vol. 1560 of *Lecture Notes in Computer Science*, Springer, pp. 53–68.

[48] Genet, T., and Klay, F. Rewriting for cryptographic protocol verification. In *Conference on Automated Deduction* (2000), D. A. McAllester, Ed., vol. 1831 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 271–290.

[49] Gilbert, H., Sibert, H., and Robshaw, M. An active attack against a provably secure lightweight authentication protocol. *IEE Electronic Letters 41*, 21 (2005), 1169–1170.

[50] Goldreich, O. *Foundations of Cryptography*. Cambridge University Press, 2000, ch. 2.2.4.2, p. 41.

[51] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game. In *Symposium on Theory of Computation – STOC* (January 1987), ACM Press, pp. 218–229.

[52] Goldschlag, D. M., Reed, M. G., and Syverson, P. F. Hiding routing information. In *Informatin Hiding* (1996), R. J. Anderson, Ed., vol. 1174 of *Lecture Notes in Computer Science*, Springer, pp. 137–150.

[53] Golle, P., Jakobsson, M., Juels, A., and Syverson, P. Universal re-encryption for mixnets. In *RSA Conference Cryptographers' Track – RSA-CT* (2004), T. Okamoto, Ed., vol. 2964, Springer, pp. 163–178.

[54] Greentech Computing. Gt6 algorithm solves the extended DIMACS 32-bit parity problem. Tech. rep., Greentech Computing Limited, London, England, 1998.

[55] Groth, J., Ostrovsky, R., and Sahai, A. Perfect non-interactive zero knowledge for NP. In *Advances in Cryptology – EUROCRYPT '06* (To appear 2006), Springer.

[56] Håstad, J. Some optimal inapproximability results. In *Symposium on Theory of Computation – STOC* (1997), P. Shor, Ed., ACM Press, pp. 1–10.

[57] Henrici, D., and Müller, P. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *Pervasive Computing and Communication Security* (2004), IEEE Press, pp. 149–153.

[58] Hoffstein, J., Pipher, J., and Silverman, J. H. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory* (1998), J. Buhler, Ed., vol. 1423 of *Lecture Notes in Computer Science*, Springer, pp. 267–288.

[59] Hohenberger, S., and Weis, S. A. Honest-verifier private disjointness testing. In *Workshop on Privacy Enhahcing Technologies* (To appear June 2006), G. Danezis and P. Golle, Eds., Springer.

[60] Holman, W. T., Connelly, J. A., and Dowlatabadi, A. An integrated analog/digital random noise source. *IEEE Transactions on Circuits and Systems 44*, 6 (June 1997), 521–527.

[61] Hopper, N., and Blum, M. A secure human-computer authentication scheme. Tech. Rep. CMU-CS-00-139, Carnegie Mellon University, 2000.

[62] Hopper, N. J., and Blum, M. Secure human identification protocols. In *Advances in Cryptology – ASIACRYPT '01* (2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*, Springer, pp. 52–66.

[63] Johnson, D. S., and Trick, M. A., Eds. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge* (1993), vol. 26, American Mathematical Society.

[64] Juels, A. Minimalist cryptography for RFID tags. In *Security in Communication Networks* (2004), C. Blundo and S. Cimato, Eds., vol. 3352 of *Lecture Notes in Computer Science*, Springer, pp. 149–164.

[65] Juels, A. "Yoking proofs" for RFID tags. In *Pervasive Computing and Communication Security* (2004), IEEE Press.

[66] Juels, A., and Pappu, R. Squealing Euros: Privacy protection in RFID-enabled banknotes. In *Financial Cryptography* (2003), R. N. Wright, Ed., vol. 2742 of *Lecture Notes in Computer Science*, Springer, pp. 103–121.

[67] Juels, A., and Weis, S. A. Authenticating pervasive devices with human protocols. In *Advances in Cryptology – CRYPTO '05* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 293–308.

[68] Juels, A., and Weis, S. A. Defining strong privacy for RFID. Submitted for Publication, 2006.

[69] Katz, J., and Shin, J. S. Parallel and concurrent security of the HB and HB+ protocols. In *Advances in Cryptology – EUROCRYPT '06* (To appear May 2006), S. Vaudenay, Ed., Lecture Notes in Computer Science, Springer.

[70] Kearns, M. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM 45*, 6 (November 1998), 983–1006.

[71] Kiayias, A., and Mitrofanova, A. Testing disjointness of private datasets. In *Financial Cryptography* (2005), A. S. Patrick and M. Yung, Eds., vol. 3570 of *Lecture Notes in Computer Science*, Springer, pp. 109–124.

[72] Kissner, L., and Song, D. Privacy-preserving set operations. In *Advances in Cryptology – CRYPTO '05* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 241–257.

[73] Kohlbrenner, P., and Gaj, K. An embedded true random number generator for FPGAs. In *Symposium on Field Programmable Gate Arrays* (2004), H. Schmit, Ed., ACM Press, pp. 71–78.

[74] Levieil, E., and Fouque, P.-A. An attack of HB+ in the detection-based model. In submission, To appear 2006.

[75] Lindell, Y., and Pinkas, B. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO '00* (2000), M. Bellare, Ed., vol. 1880 of *Lecture Notes in Computer Science*, Springer, pp. 20–24.

[76] MacWilliams, F., and Sloane, N. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

[77] Mandel, J., Roach, A., and Winstein, K. MIT Proximity card vulnerabilities. Available at http://web.mit.edu/keithw/Public/MIT-Card-Vulnerabilities-March31.pdf, March 2004.

[78] Massey, J. L., and Omura, J. K. A new multiplicative algorithm over finite fields and its applicability in public key cryptography. Presented at rump session of EUROCRYPT '83, March 1983.

[79] Matsumoto, T. Human-computer cryptography: An attempt. In *Computer and Communications Security – CCS* (1996), ACM Press, pp. 68–75.

[80] Matsumoto, T., and Imai, H. Human identification through insecure channel. In *Advances in Cryptology – EUROCRYPT '91* (1991), D. W. Davies, Ed., vol. 547 of *Lecture Notes in Computer Science*, Springer, pp. 409–421.

[81] McEliece, R. J. DSN Progress Report. Tech. Rep. 42–44, JPL-Caltech, 1978.

[82] Meadows, C. A. Formal verification of cryptographic protocols: A survey. In *Advances in Cryptology – ASIACRYPT '95* (1995), J. Pieprzyk and R. Safavi-Naini, Eds., vol. 917 of *Lecture Notes in Computer Science*, Springer, pp. 135–150.

[83] Meadows, C. A. Open issues in formal methods for cryptographic protocol analysis. In *Methods, Models, and Architectures for Network Security* (2001), V. I. Gorodetski, V. A. Skormin, and L. J. Popyack, Eds., vol. 2052 of *Lecture Notes in Computer Science*, Springer, p. 21.

[84] Metcalfe, R. M., and Boggs, D. R. Ethernet: Distributed packet switching ethernet: Distributed packet switching for local computer networks. *Communications of the ACM 19*, 5 (July 1976), 395–404.

[85] Micali, S., Rabin, M., and Kilian, J. Zero-knowledge sets. In *Foundations of Computer Science – FOCS* (2003), M. Sudan, Ed., IEEE Press, pp. 80–91.

[86] Millen, J. On the freedom of decryption. *Information Processing Letters 86*, 6 (June 2003), 329–333.

[87] Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review 63* (1956), 81–97.

[88] Molnar, D., and Wagner, D. Privacy and security in library RFID : Issues, practices, and architectures. In *Computer and Communications Security – CCS* (2004), ACM Press, pp. 210 – 219.

[89] Naor, M., and Pinkas, B. Visual authentication and identification. In *Advances in Cryptology – CRYPTO '97* (1997), B. S. K. Jr., Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer, pp. 322–336.

[90] Naor, M., and Pinkas, B. Oblivious transfer and polynomial evaluation. In *Symposium on Theory of Computation – STOC* (May 1999), T. Leighton, Ed., ACM Press, pp. 245–254.

[91] Niederreiter, H. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory 15*, 2 (1986), 159–166.

[92] Ohkubo, M., Suzuki, K., and Kinoshita, S. Efficient hash-chain based RFID privacy protection scheme. In *Ubiquitous Computing (UBICOMP)* (September 2004).

[93] Oleshchuk, V. A. On public-key cryptosystems based on Church-Rosser string-rewriting systems. In *COCOON '95* (1995), J. Calmet and J. A. Campbell, Eds., vol. 959 of *Lecture Notes in Computer Science*, Springer, pp. 264–269.

[94] Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT '99* (May 1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 223–238.

[95] Paillier, P. Trapdooring discrete logarithms on elliptic curves over rings. In *Advances in Cryptology – ASIACRYPT '00* (2000), T. Okamoto, Ed., vol. 1976 of *Lecture Notes in Computer Science*, Springer, pp. 573–584.

[96] Pedersen, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91* (1991), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer, pp. 129–140.

[97] Petrie, C. S., and Connelly, J. A. A noise-based IC random number generator for applications in cryptography. *IEEE Transactions on Circuits and Systems 47*, 5 (May 2000), 615–620.

[98] Pinkas, B. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKKD Explorations Newsletter 4*, 2 (December 2002), 12–19.

[99] Pohlig, S. C., and Hellman, M. E. An improved algorithm for computing logarithms over GF($P$) and its cryptographic significance. *IEEE Transactions on Information Theory IT-24* (1978), 106–110.

[100] Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *Symposium on Theory of Computation – STOC* (2005), R. Fagin, Ed., ACM Press, pp. 84–93.

[101] Rivas, M. RFID technology and applications. In *MIT RFID Privacy Workshop* (November 2003).

[102] Rivest, R., and Weis, S. A. Commutative and cascadable cryptography. Submitted for Publication, 2006.

[103] Samuel, S. C., Thomas, D. G., Abisha, P. J., and Subramanian, K. G. Tree replacement and public key cryptosystem. In *Progress in Cryptology - INDOCRYPT '02* (2002), A. Menezes and P. Sarkar, Eds., vol. 2551 of *Lecture Notes in Computer Science*, Springer, pp. 71–78.

[104] Sarma, S. E., Weis, S. A., and Engels, D. W. RFID systems and security and privacy implications. In *Workshop on Cryptographic Hardware and Embedded Systems* (2002), B. S. Kaliski, Çetin Kaya Koç, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, Springer, pp. 454–470.

[105] Shamir, A. On the power of commutativity in cryptography. Presented at ICALP'80, July 1980.

[106] Shamir, A., Rivest, R. L., and Adleman, L. M. *The Mathematical Gardner*. David A. Klarner, Ed., Wadsworth International, 1981, ch. Mental Poker, pp. 37–43.

[107] Shamos, M. I. Paper v. electronic voting records - An assessment. Available at: http://euro.ecom.cmu.edu/people/faculty/mshamos/paper.htm, April 2004.

[108] Stern, J. A new paradigm for public key identification. In *Advances in Cryptology – CRYPTO '93* (1993), D. R. Stinson, Ed., vol. 773 of *Lecture Notes in Computer Science*, Springer, pp. 13–21.

[109] Stern, J. A new paradigm for public key identification. *IEEE Transactions on Information Theory 42*, 6 (1996), 1757–1768.

[110] Subramanian, V., Chang, P. C., Huang, D., Lee, J. B., Molesa, S. E., Redinger, D. R., and Volkman, S. K. All-printed RFID tags: Materials, devices, and circuit implications. In *VLSI Design* (2006), IEEE Press, pp. 709–714.

[111] Subramanian, V., Chang, P. C., Lee, J. B., Molesa, S. E., and Volkman, S. K. Printed organic transistors for ultra-low-cost RFID applications. *IEEE Transactions on Components and Packaging Technology 28*, 4 (2004), 742–747.

[112] Tomassini, M., Sipper, M., and Perrenoud, M. On the generation of high quality random numbers by two-dimensional cellular automata. *IEEE Transactions on Computers 49*, 10 (2000), 1146–1151.

[113] United Nations. Convention against torture and other cruel, inhuman or degrading treatment or punishment, December 1984.

[114] United States Department of State. The U.S. Electronic Passport. http://travel.state.gov/passport/eppt/eppt_2498.html, March 2006.

[115] Verichip. Website. http://www.4verichip.com/, 2005.

[116] Wagner, D. A generalized birthday problem. In *Advances in Cryptology – CRYPTO'02* (2002), M. Yung, Ed., vol. 2442 of *Lecture Notes in Computer Science*, Springer, pp. 288–303.

[117] Wagner, N. R., and Magyari, M. R. A public-key cryptosystem based on the word problem. In *Advances in Cryptology – CRYPTO '84* (1984), G. R. Blakley and D. Chaum, Eds., vol. 196 of *Lecture Notes in Computer Science*, Springer, pp. 19–36.

[118] Wang, C.-H., Hwang, T., and Tsai, J.-J. On the Matsumoto and Imai's human identification scheme. In *Advances in Cryptology – EUROCRYPT '95* (1995), L. C. Guillou and J.-J. Quisquater, Eds., vol. 921 of *Lecture Notes in Computer Science*, Springer, pp. 382–392.

[119] Warners, J. P., and Van Maaren, H. A two phase algorithm for solving a class of hard satisfiability problems. *Operations Research Letters 23*, 3–5 (1999), 81–88.

[120] Weis, S. A. Security and privacy in radio-frequency identification devices. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, May 2003.

[121] Weis, S. A. Security parallels between people and pervasive devices. In *Pervasive Computing and Communication Security* (2005), IEEE Press. To appear.

[122] Weis, S. A., Sarma, S. E., Rivest, R. L., and Engels, D. W. Security and privacy aspects of low-cost radio frequency identification systems. In *Security in Pervasive Computing* (2003), D. Hutter, G. Müller, W. Stephan, and M. Ullmann, Eds., vol. 2802 of *Lecture Notes in Computer Science*, Springer, pp. 201–212.

[123] Wired News. Wal-mart, DoD forcing RFID. Available at http://www.wired.com/news/business/0,1367,61059,00.html, November 2003.

[124] Yamamura, A., and Saito, T. Private information retrieval based on the private information retrieval based on subgroup membership problem. In *Australasian Conference on Information Security and Privacy* (2001), V. Varadharajan and Y. Mu, Eds., vol. 2119 of *Lecture Notes in Computer Science*, Springer, pp. 206–220.

[125] Yao, A. C. How to generate and exchange secrets. In *Foundations of Computer Science – FOCS* (1986), IEEE Press, pp. 162–167.